Expert Systems With Applications 135 (2019) 12–38

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Analysis of the proficiency of fully connected neural networks in the process of classifying digital images. Benchmark of different classification algorithms on high-level image features from convolutional layers

Jonathan Janke[a], Mauro Castelli[a,*], Aleš Popovič[b,a]

[a] Nova Information Management School (NOVA IMS), Universidade Nova de Lisboa. Lisboa, 1070-312, Portugal
[b] Faculty of Economics, University of Ljubljana. Kardeljeva ploščad 17, Ljubljana 1000, Slovenia

ARTICLE INFO

Article history:
Received 20 January 2019
Revised 27 May 2019
Accepted 31 May 2019
Available online 31 May 2019

Keywords:
Convolutional neural networks
Computer vision
Image classification

ABSTRACT

Over the course of research on convolutional neural network (CNN) architectures, few modifications have been made to the fully connected layers at the ends of the networks. In image classification, these neural network layers are responsible for creating the final classification results based on the output of the last layer of high-level image filters. Before the breakthrough of CNNs, these image filters were handcrafted, and any classification algorithm could be applied to their output. Because neural networks use gradient descent to learn their weights subject to the classification error, fully connected neural networks are a natural choice for CNNs. But a question arises: Are fully connected layers in a CNN superior to other classification algorithms? In this work, we benchmark different classification algorithms on CNNs by removing the existing fully connected classifiers. Thus, the flattened output from the last convolutional layer is used as the input for multiple benchmark classification algorithms. To ensure the generalisability of the findings, numerous CNNs are trained on CIFAR-10, CIFAR-100, and a subset of ILSVRC-2012 with 100 classes. The experimental results reveal that multiple classification algorithms, namely logistic regression, support vector machines, eXtreme gradient boosting, random forests and K-nearest neighbours, are capable of outperforming fully connected neural networks. Furthermore, the superiority of a particular classification algorithm depends on the underlying CNN structure and the nature of the classification problem. For classification problems with many classes or for CNNs that produce many high-level image features, other classification algorithms are likely to perform better than fully connected neural networks. It follows that it is advisable to benchmark multiple classification algorithms on high-level image features produced from the CNN layers to improve classification performance.

1. Introduction

Computer vision is a sub-field of artificial intelligence and computer science that enables computers to develop a visual perception of real-world entities (Szeliski, 2010). This is achieved by automatically extracting, analysing, and understanding information from input images. The field of computer vision can be divided into multiple sub-areas, each of which focuses on specific information of the image data: classification, localisation, detection, semantic segmentation, and instance segmentation (see Fig. 1).

This paper focuses on algorithms applied to the task of image classification.

Before the dominance of neural networks in computer vision research, any classification algorithm was used to distinguish the classes based on the output from manually designed feature extractors (filters) (LeCun, Bottou, Bengio, & Haffner, 1998). The emergence of convolutional neural networks in computer vision produced a shift from hand-designed feature extractors to automatically generated feature extractors trained with backpropagation.

Computer vision has made a lot of progress since the breakthrough of artificial neural networks (ANN). This development was fostered by the increases in available computational power and training data. But it was only in 2012, when a research team from the University of Toronto constructed AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) for the ImageNet Large Scale Visual

* Corresponding author.
E-mail addresses: jonathan.janke@ghcs.de (J. Janke), mcastelli@novaims.unl.pt (M. Castelli), ales.popovic@ef.uni-lj.si (A. Popovič).

https://doi.org/10.1016/j.eswa.2019.05.058
0957-4174/© 2019 Elsevier Ltd. All rights reserved.

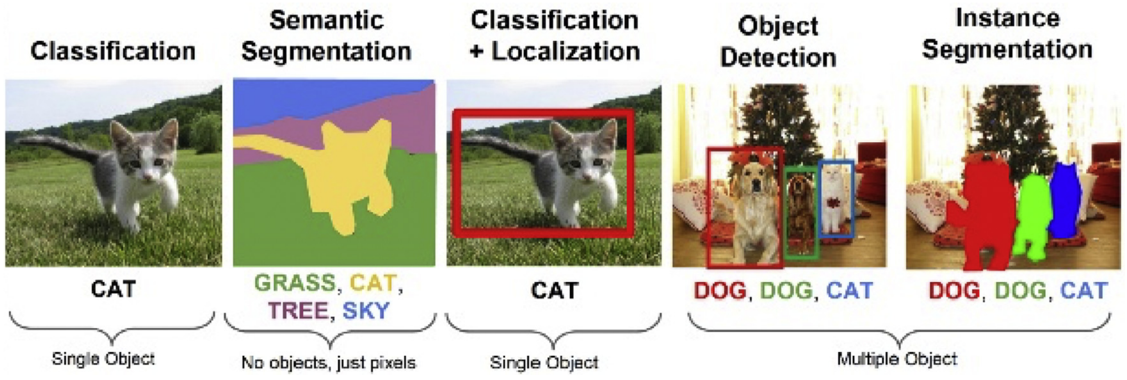


Fig. 1. Summary of different computer vision tasks from (Li Fei-Fei & Yeung, 2017).

Recognition Challenge (ILSVRC-2012), that a convolutional neural network (CNN) architecture outperformed traditional approaches in the classification and localisation tasks (Russakovsky et al., 2015). After that, CNN architectures were improved in many ways, found their way into multiple fields of research and were successfully applied in the industry (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016).

In recent years, there have been many developments in CNN architectures (Simonyan & Zisserman, 2014). The vast majority of these advances have been made in the earlier layers of the neural network, meaning everything up to the fully connected layers. Little research has been devoted to the proficiency of the fully connected layers. Fully connected multi-layer perceptrons (MLPs) seem to be the design choice for all network architectures, with little research on or modifications made to these layers. But as the no-free-lunch theorem states (Wolpert & Macready, 1997), no classifier can be said to be superior to all other classifiers in every problem instance. Therefore, following the idea of the traditional model of pattern recognition wherein any classification algorithm was used, the main research question for this paper is:

Research question ”Is it possible to improve the performance of the computer vision classification model by using different classification algorithms on high-level image features?”

Both convolutional and fully connected layers of a neural network learn their weights during the training phase, with the error being backpropagated from the output through the fully connected-layers back to the convolutional layers. Due to the nature of the learning process, multi-layer perceptrons are a natural choice for convolutional neural networks. Therefore, the focus of this paper is not directly on replacing the fully connected layers in the learning process, but rather on enhancing the learning process by using a two-step procedure:

1. **Regular CNN training process:** Train a given CNN architecture, including fully connected layers to make convolutional layers learn high-level image features from image inputs.
2. **Enhancement of the training process:** Replace the fully connected layers with a different classification algorithm and train this algorithm based on the high-level image features produced from the last layer of convolutional layers in the first step.

The research question is tested over multiple settings and multiple datasets to fully investigate under which conditions certain classification algorithms perform better. The datasets benchmarked in this paper are CIFAR-10, CIFAR-100, and a subset of ILSVRC-2012.

The paper is organized as follows: Section 2 gives an overview of related work in the field of CNNs. Afterwards, Section 3 outlines the experimental setup used to test the research question proposed above, including the datasets, the network architectures, and the chosen classification algorithms. Section 4 discusses the experimental results and provides some guidelines for the choice of the classification algorithm to be used for classifying images by using the high-level image features. Finally, Section 5 concludes the paper by summarising the main findings of this study and suggesting future avenues of research.

2. Related works

The ability of multilayer networks trained via gradient descent to learn complex, high-dimensional, nonlinear mappings from large collections of examples makes them obvious candidates for image recognition tasks (LeCun et al., 1998).

This notion of learning is consistent with the general learning process of the human brain, because CNNs automatically detect two-dimensional local structures within images without prior manual definition. In comparison to fully-connected feedforward neural networks, CNNs are also computationally more efficient because the convolutional network layers require fewer weights. Thus, the model has to learn fewer parameters during the training process (Krizhevsky et al., 2012).

The first published CNN, LeNet-5, was developed by LeCun et al. (1998) and was applied in alphanumeric character recognition to read “several million checks per day” (LeCun et al., 1998, p. 1) automatically.

After their introduction, CNNs remained at the sidelines of research until AlexNet won the ILSVRC-2012 challenge in the classification and localisation task by a great margin compared to the other approaches (Russakovsky et al., 2015). For the classification task, AlexNet reached a top-5 error of 15.32%, whereas the next best model architecture reached 26.17%. AlexNet (Krizhevsky et al., 2012) is an 8-layer neural network that strongly resembles LeNet-5 from Fig. 2 in its architecture. After this success, CNNs became the state of the art in computer vision, with more and more innovative additions to the networks outperforming earlier models. The next decisive changes to the network architecture were introduced with GoogLeNet from Szegedy et al. (2015) and VGG from Simonyan and Zisserman (2014). Both CNNs competed in the ILSVRC-2014 competition, reaching a top-5 classification error of 6.67% and 7.32% respectively (Russakovsky et al., 2015; Szegedy et al., 2015). The main contribution of VGG was to use relatively small convolutional filters combined with deeper neural networks (16–19 layers) (Simonyan & Zisserman, 2014). GoogLeNet is a 22-layer network without fully-connected layers at its end. Its main contribution to subsequent model architectures is its inception module. The inception module performs multiple parallel convolutions and enables the model to determine which convolutional layer is proficient (Szegedy et al., 2015). Apart from computational limitations, a roadblock to even deeper neural network architectures was the

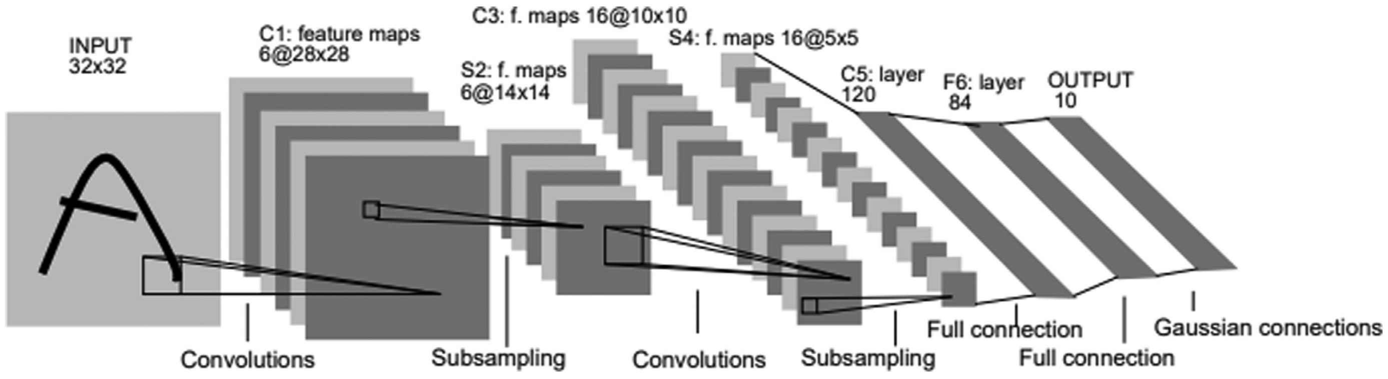


Fig. 2. LeNet-5 architecture from LeCun et al. (1998).

fact that deeper networks empirically produced worse results on training and test data (He, Zhang, Ren, & Sun, 2016). They should, in theory, be at least as good as their shallower counterparts. The ResNet-152 architecture (He et al., 2016) won the ILSVRC-2015 challenge with a top-5 error of 3.57%. Thus, it was the first model to beat human performance. The solution to training deeper networks with 152 layers are residual blocks, which lent their name to ResNet. Residual blocks enable connections between multiple non-adjacent layers such that the layers in between them only need to learn the residuals (He et al., 2016).

Further research has shown that the computationally expensive last fully-connected layer can be set to constant values to reduce model complexity with little or no performance loss (Hoffer, Hubara, & Soudry, 2018). To illustrate their example, the researchers show that about 60% of the 36 million model parameters of a ResNet-50 model reside in the last fully-connected layer for the JFT-300M dataset[1] (Hoffer et al., 2018).

Another interesting approach to reduce the complexity of the model architectures has been developed by HasanPour, Rouhani, Fayyaz, and Sabokrou (2016), in which the authors showed that they can perform on par to very large and deep neural networks with significantly fewer parameters and smaller model sizes. They achieved these results by applying a set of defined design principles. They proved that their networks perform similarly to state-of-the-art[2] CNN architectures on various well-known datasets, e.g., MNIST (LeCun & Cortes, 2010), CIFAR-100 (Krizhevsky & Hinton, 2009), CIFAR-100 (Krizhevsky & Hinton, 2009), and ILSVRC-2012 (Russakovsky et al., 2015).

2.1. Literature review

This section presents related works where a CNN was used as a feature extractor, and where the features extracted by the CNN were subsequently used by a classifier to determine the correct label of an image. The main working principle of a CNN is to extract high-level features from an image by means of a sequence of convolutional and max-pooling layers. Several methods were defined in recent years for improving the performance of the standard CNN architecture. This section aims at introducing existing works related to the method proposed in this paper.

Chen, Jiang, Li, Jia, and Ghamisi (2016) use a deep feature extraction method for hyperspectral images (HSI). The authors propose three deep feature extraction architectures, based on CNNs, to extract the spectral, spatial, and spectral-spatial features of HSI. The authors define a 3-D CNN to capture different kinds of spectral-spatial features. The subsequent classification task is performed by taking into account a fully-connected neural network and by comparing its performance with the one achieved by logistic regression, support vector machines (SVMs), and K-nearest neighbors. The main reason for considering other classifiers is the lack of sufficient training data. The authors show that the designed 3-D CNN can extract the spectral-spatial features effectively, which leads to satisfactory classification performance.

A similar approach is presented in Liu, Sun, Meng, Wang, and Fu (2018), where the authors propose a 3D-CNN for HSI classification. To solve the problem of insufficient samples of HSI, they also introduce a technique for generating virtual samples. The experimental results show the suitability of the 3D-CNN for addressing the classification problem at hand, while the generation of virtual samples is beneficial for improving the performance of the CNN-based model over unseen data (generalisation). The final HSI classification from the image features is performed by means of a neural network.

Ding, Li, Hu, Zhang, and Wang (2018) present an approach to make predictions based on deep features extracted from AlexNet. Apart from the AlexNet features, the authors include spectral features and gray level co-occurrence matrix (GLCM) texture features into the prediction process. The analysis consequently focuses on the proficiency and predictive capability of these feature sets and puts less emphasis on the SVM used as a predictor. The authors come to the conclusion that the CNN features are much more useful than the other features analyzed.

In Yang, Zhao, and Chan (2017), the authors propose a two-branch CNN as a feature extractor. The two branches of the CNN are responsible for extracting features from the spectral domain as well as the spatial domain. The set of learned features is then used for extracting the joint spectral-spatial features for classification. To improve the classification performance of the fully-connected neural network the use of a pretrained network was also considered. A similar approach was recently presented by Ma, Fu, Wang, Wang, and Yin (2018). The authors designed a two-branch CNN with a skip architecture to learn the spectral-spatial features. In the first branch, a band selection layer is designed to reduce parameters and limit overfitting. Unpooling and deconvolution operations are utilized to recreate the lost information from the pooling layers, while the skip architecture allows the deep semantic information to merge with the shallow appearance information. The second branch is responsible for learning deep spectral features. Experimental results show the suitability of the proposed method, as it reduces overfitting and obtains a competitive classification performance. Nonetheless, as in Yang et al. (2017), this increase in robustness comes at the cost of longer training time.

In Chen, Lin, Zhao, Wang, and Gu (2014), the authors propose a hybrid approach for the classification of images. The framework

[1] JFT-300M is an internal dataset from Google with more than 18k different classes in 300 million images, see also Sun, Shrivastava, Singh, and Gupta (2017).

[2] As of August 2016.

consists of a combination of principle component analysis (PCA), deep learning architecture, and logistic regression. In particular, the authors consider stacked autoencoders as the deep learning architecture for extracting the images' features. The experimental results demonstrate that the proposed hybrid framework produces competitive results over well-known datasets. The idea of using a hybrid method for feature extraction is also pursued in the work of Romero, Gatta, and Camps-Valls (2016), where a CNN is combined with an algorithm for unsupervised learning of features. Experimental results show the superiority of the proposed method with respect to standard machine learning procedures for image classification. In more detail, results demonstrate that single-layer convolutional networks can extract powerful discriminative features, but deep architectures significantly outperform single-layer variants capturing increasing levels of abstraction and complexity throughout the feature hierarchy. Another hybrid method is proposed in Cao et al. (2018), where the authors formulate the image classification problem from a Bayesian perspective and address it by means of Markov random fields and CNNs. In particular, the CNN layers are used for learning the posterior class distributions using a patch-wise training strategy.

A recent semi-supervised approach for addressing image classification tasks in semi-supervised environments was proposed by Ling, Li, Zou, and Guo (2018). The authors propose the use of CNNs with a modified loss function. Additionally to the conventional softmax loss function for labeled data, they integrate a K-means clustering loss function for unlabeled data. In this case, the labeled features extracted from a CNN are not only used for training the classifier, but also provide the anchors to initialize a set of clustering centers used by the K-means algorithm. Experimental results show the suitability of this method for addressing the problems in which collecting labeled data is an expensive or time-consuming task. Another semi-supervised approach is presented in Liu et al. (2017), where the authors proposed a semi-supervised network with a skip connection between the encoder and the decoder in order to solve the problem of limited labeled samples.

The works described in Hertel, Barth, Käster, and Martinetz (2017) and in Ayinde, Inanc, and Zurada (2019) perform a more general analysis of CNNs being generic feature extractors. Both papers only look at the question of transfer learning using neural networks. They do not investigate the usage of other classification algorithms based on the image features. In Hertel et al. (2017), the authors show that CNNs are proficient in transfer learning environments where only the final classification part must be retrained. They use pretrained CNN models and keep the convolutional layers constant, and they only retrain the final fully-connected layers. They show the suitability of CNNs in transfer learning by outperforming the benchmark on the CIFAR-100 dataset through this approach. Furthermore, this inevitably enables lower costs for training the networks. In Ayinde et al. (2019), the authors focus their research on the properties of these extracted features, without analyzing their predictive capabilities.

With respect to the study presented in this paper, all the aforementioned contributions propose the use of CNNs, with or without dimensionality reduction techniques, for extracting the salient high-level features of the images. Anyway, these methods do no put an emphasis on the last layers of CNNs, namely the fully-connected layers that are responsible for performing the classification of the input image based on the features extracted using the convolutional and pooling layers.

A first step towards the investigation of the classification performance of a CNN when the fully-connected layer is replaced with a different classifier appeared in Tang (2013). The authors replace the Softmax layer with a linear support vector machine. In this case, the learning process minimizes a margin-based loss instead of the cross-entropy loss. Thus, the authors used the loss from L2-SVM instead of the standard and linear hinge loss. Experimental results show the suitability of the proposed method over popular datasets.

Following the same research line, Alipourfard, Arefi, and Mahmoudi (2018) use a CNN for extracting image features. Subsequently, they apply a dimensionality reduction technique for reducing the number of features. They use the reduced set of features to perform a simple binary classification by means of a logistic regression model. Experimental results are competitive with respect to the ones published in the existing literature. Nonetheless, improvement in terms of classification performance is mostly attributed to the dimensionality reduction method.

In Lu, Yuan, and Fang (2017), the authors discuss the applicability of convolutional neural networks as feature extractors in the area of aerial scene classification. Instead of using fully-connected layers to make predictions from the image features, the final classification is performed with an SVM. In more detail, the output features are clustered and the mapping of cluster centroids to the corresponding classes is approximated using an SVM. Aerial scene classification is a specific learning environment that is characterized by the sparsity of available training images. Therefore, the authors' main motivation to replace the fully-connected layer is to reduce the number of parameters that the model needs to learn. Convolutional neural networks have to learn a lot of parameters and, proportionally, most are in the fully-connected layers. Therefore, a reasonable approach is to reduce the complexity of the network to cluster the outputs and use a shallow classifier to make this prediction. As a result, the authors found that their approach produces comparable results to other state-of-the-art approaches in this particular area, including fully-connected neural networks.

In Alaslani and Elrefaei (2018), the authors use AlexNet to extract high-level features associated with iris images and perform a classification using an SVM. Although they mention that "different types of classifiers can be used for this task, for example, Support Vector Machine, Softmax Regression, and Neural Network" (Alaslani & Elrefaei, 2018, p. 72), they only try out SVMs in the experimental phase. Their research is limited to relatively small datasets, consisting of between 7 to 10 samples per class. In this environment, neural network based models are usually not proficient due to the lack of training data. Regarding the deep feature extraction, the authors try out using features from different layers of the neural networks. They extract the features before the final classification but not necessarily before the fully-connected layers. A similar approach is proposed by Minaee, Abdolrashidi, and Wang (2017). The authors use VGG-Net to extract the image features and also use an SVM for the final classification, showing a satisfactory performance in terms of accuracy.

The work presented in Bodapati and Veeranjaneyulu (2019) apply a similar approach with respect to the one presented in our paper. The authors use a deep convolutional neural network that they define for the feature extraction, and, subsequently, they perform a classification using an SVM. They do not compare their approach with other classification algorithms, apart from fully-connected layers as the benchmark for a traditional full CNN approach. They come to the conclusion that the separate approach using SVMs performs similar or slightly better to the traditional CNN approach with fully-connected layers.

Following a similar approach in Thomaz, Carneiro, and Patrocinio (2017), the authors try out a separate learning approach by using a CNN as a feature extractor and then making a classification using a multilayer perceptron. Their research is based on a breast cancer dataset that consists of only 307 images. They use a separate learning procedure of using a CNN and extracting the image features and then forwarding those features to a new neural network. Anyway, as pointed out by the authors, the results obtained are not robust due to the small amount of data and the memory constraints they faced. Thus, they highlight the need for a deeper analysis that also considers different classifiers. Corchs, Fersini, and Gasparini (2017) adopted an ensemble learning approach based on the Bayesian model averaging method that combines five state-of-the-art classifiers. The proposed method considers predictions given by several classification models, based on visual and textual data, through a late and an early fusion scheme, respectively. Experimental results show that an ensemble method based on a late fusion of unimodal classifiers permits high classification performance achievement with respect to a traditional CNN. An ensemble learning approach was also proposed by Srivastava, Mukherjee, Lall, and Jaiswal (2017). In this work, the authors propose an ensemble of local and deep features for object classification. In particular, CNNs are used for extracting different features from the images, and these features are finally processed by an ensemble of support vector machines. As commonly done in ensemble learning, the final classification is obtained by considering a majority vote obtained from the pool of SVMs.

To the best of our knowledge, the most recent approach that poses an alternative to traditional CNNs is the method proposed in Zhang et al. (2019). In this paper, the authors propose the use of capsule networks (Sabour, Frosst, & Hinton, 2017) for image classification, showing better performance with respect to traditional CNNs for the dataset taken into account. Nonetheless, capsule networks are a conceptually different network structure than convolutional neural networks, so this approach cannot be compared to the approach from this paper.

To provide the reader with an overview of the existing contributions in this area, Table 1 summarizes the content of this section by highlighting the salient points of the papers cited. The analysis of the literature review demonstrated the need for exhaustive studies aimed at testing the predictive capability of different algorithms on high-level image features. This is a fundamental step towards a better understanding of the properties of CNNs.

2.2. Strengths and weaknesses of proposed method

In comparison to the other approaches presented in Section 2.1, this paper conducts an exhaustive study of multiple algorithms to test their predictive capability on high-level image features. This section takes a more in-depth look at the strengths and weaknesses of our proposed method in comparison to the literature presented in Section 2.1.

Some of the other authors use Convolutional Neural Networks as feature extractors to perform a classification using other classification algorithms than a fully-connected neural network, e.g. Chen et al. (2016), Liu et al. (2018), Chen et al. (2014), Srivastava et al. (2017), Corchs et al. (2017), Bodapati and Veeranjaneyulu (2019), Minaee et al. (2017), Alaslani and Elrefaei (2018), Lu et al. (2017), Tang (2013), Alipourfard et al. (2018) and Ding et al. (2018) (see Table 1). Nonetheless, this is mostly done with the intention of reducing the number of parameters to be optimized because the authors use datasets with few training samples. In the datasets used in this paper, the size is not a primary concern. The other approaches opt for shallower classifiers caused by the constraints of their datasets. No other approach analyzes the proficiency of using another classifier with respect to using regular fully-connected layers for large datasets. This paper takes different classification algorithms into consideration and experimentally determines their advantages in different scenarios. This enables the establishment of a guideline under which circumstances certain algorithms are more likely to outperform other algorithms, as detailed in Section 4.4.

Other than that, some authors use CNNs as universal feature extractors for transfer learning tasks, such that only the final classifier needs to be retrained, e.g., Hertel et al. (2017) and Ayinde et al. (2019). Firstly, these approaches still use neural networks in the subsequent steps to make predictions. Secondly, this paper's approach does not look at the case of transfer learning but focuses on improving the performance on the original dataset instead.

On the downside, this approach mainly comes at the cost of resources. Our proposed method did not look at the possibility of training the subsequent classifier using backpropagation. Instead, we designed a separate learning procedure. As the separate learning procedure includes the full training of a CNN to generate the image features and needs further training of a classifier, it is inevitably longer than simply training a CNN by itself. This limitation could be resolved by using a transfer learning approach or universal image features.

Finally, as discussed in more detail in Section 4, while some classifiers are able to outperform fully-connected neural networks, no classifier was consistently better than fully-connected networks. Apart from logistic regression, the performance of the other classifiers presents a greater variance than the performance of fully-connected neural networks. Thus, improving the classifier's performance comes with a greater risk and also requires the testing of different methods before deciding which is the best. Following the experimentally established guidelines in Section 4.4 likely reduces the risk of choosing a bad classifier in a given scenario.

All in all, this paper proposes a complete study that aims at analyzing the proficiency of fully-connected neural networks in classifying digital images based on high-level features extracted with a CNN. This is an important contribution to the CNN field because it experimentally shows that replacing the fully-connected neural networks with other classifiers can improve the accuracy of the model. In particular, CNN practitioners could take advantage of the guidelines that were experimentally defined when they have to address an image classification problem.

3. Experimental setting

This section presents the experimental setup, providing all the necessary information to reproduce the results discussed in Section 4.

The computation was mainly done on commodity hardware,[3] which limits the size of the datasets, the benchmark algorithms, and the hyper-parameters used for the comparison to a reasonable degree. Nonetheless, the algorithms and ideas developed are scalable and applicable on larger datasets subject to computational means.

3.1. Experimental procedure

From a technical perspective, the procedure adopted in the experimental phase is the two-step procedure from Section 1. The code used to perform the experimental campaign can be found on GitHub.[4]

The first part of the algorithm generates *intermediate* output features. These output features are produced on several image datasets: CIFAR-10, CIFAR-100 and a subset of ILSVRC-2012. For ILSVRC-2012, the models are only trained on a reduced dataset containing 100 classes instead of the original 1000 classes. Preprocessing is applied to the images. For CIFAR-10 and CIFAR-100, the images are normalised using z-scaling.[5] For ILSVRC-2012, the images are scaled between −1 and 1.

The high-level output features are generated by training different neural network architectures. Section 3.2 describes the

[3] Windows 10, Intel Core i7-4510U CPU @ 2.60 GHz, 8.00 GB RAM, GPU not used; MacBook Pro-2017, Intel Core i7 @ 2.80 GHz, 16 GB RAM, GPU not used.

[4] https://github.com/novajon/classy-conv-features.

[5] CIFAR-10: mean of 120.707 with a standard deviation of 64.15; CIFAR-100: mean of 121.936 with a standard deviation of 68.389.

Table 1
Summary of the contributions analyzed in Section 2.1.

Reference	Feature Extraction	Classification	Comments
Chen et al. (2016)	CNN was compared with the Principal Components Analysis (PCA), factor analysis (FA), and locally linear embedding (LLE) to investigate the potential of CNN as feature extractor.	Classification performed by means of traditional fully-connected layers. SVMs, KNN, and Logistic Regression are also taken into account to evaluate the classification performance achieved by using PCA, FA, and LLE as feature extractors.	Use of different classifiers and feature extractors motivated by the lack of sufficient training data.
Liu et al. (2018)	3-D CNN is used to extract the spectral-spatial feature information.	Classification performed by means of traditional fully-connected layers. Comparison with SVM highlights the importance of having a sufficient number of training observations.	Application characterized by a very limited amount of training data.
Yang et al. (2017)	A deep convolutional neural network with two-branch architecture is proposed to extract the joint spectral-spatial features from HSIs. The two branches of the proposed network are devoted to features from the spectral domain as well as the spatial domain.	Classification performed by means of traditional fully-connected layers.	Application characterized by a very limited amount of training data. The authors showed that transfer learning is beneficial for improving classification performance.
Ma et al. (2018)	The paper designs an end-to-end deconvolution network with skip architecture to learn the spectral-spatial features.	Classification performed by means of traditional fully-connected layers.	The design of the network allows to reduce overfitting. Additionally, to take advantage of information from the lower layers, the skip architecture is a viable choice.
Chen et al. (2014)	Stacked autoencoders used for extracting the high-level features.	A single layer of neurons is used for performing the classification task. The performance are compared against SVM showing a greater accuracy.	This is one of the first studies in which Deep Learning was used for extracting features from images. Several design choices look quite obsolete nowadays.
Romero et al. (2016)	Greedy layer-wise unsupervised pretraining (see the paper for the details) coupled with an algorithm for unsupervised learning of sparse features.	Classification performed by means of traditional fully-connected layers.	The proposed feature extracting algorithm outperforms standard principal component analysis (PCA) and its kernel counterpart (kPCA). Additionally, the authors show that deep architectures significantly outperform single-layer variants, capturing increasing levels of abstraction and complexity throughout the feature hierarchy.
Cao et al. (2018)	CNN with a smooth Markov Random Field (MRF) prior. The CNN is used to extract spectral-spatial features from 3D patches, and a smooth MRF prior is placed on the labels to further exploit spatial information.	Classification performed by means of traditional fully-connected layers.	This is the first contribution that formulates the image classification task in a Bayesian framework, where deep learning and MRF are considered simultaneously. Results obtained outperformed the ones achieved by SVMs and three deep learning architectures.
Ling et al. (2018)	CNN with four convolutional layers.	Classification is performed using a semi-supervised method: for labelled images the traditional architecture with fully-connected layers is used, while a K-means clustering algorithm is jointly used with the information about the labelled images to perform the classification of unlabeled images.	The semi-supervised method outperforms the accuracy obtained with a traditional SVM-based classifier as well as the accuracy achieved by state-of-the-art deep learning architectures.
Liu et al. (2017)	CNN with skip connections (Valpola, 2015).	Classification performed by means of traditional fully-connected layers. An encoder is used to assign a ground-truth label to all of the unlabeled images before the classification step.	In order to deal with limited labeled samples, the CNN is trained by semi-supervised method to simultaneously minimize the sum of supervised and unsupervised cost functions. The architecture outperforms other deep learning architectures and an SVM classifier.
Alipourfard et al. (2018)	To overcome the curse of dimensionality a subspace-based feature extraction method is performed by calculating the orthonormal basis of the correlation matrix for each class to reduce the dimensionality of the hyperspectral images and increasing the signal to noise ratio. The CNN architecture and subspace reduction method are jointly used to extract image features.	Logistic regression.	Experimental results from two real and well-known hyperspectral images show that the proposed strategy leads to performance improvement, as opposed to using the original data and conventional feature extraction strategies. This is a suitable approach when limited training samples are available.
Tang (2013)	CNN with some modifications to the standard backpropagation algorithm are introduced to make the final prediction with a linear SVM.	Linear support vector machines.	The authors show that by simply replacing softmax with linear SVMs gives significant gains on popular deep learning datasets MNIST, CIFAR-10, and the ICML 2013 Representation Learning Workshop's face expression recognition challenge.
Lu et al. (2017)	CNN that has different size of convolutional kernels in the same layer and ignores the fully convolutional layer, so it has fewer parameters and can be trained well on small training sets.	The fully-connected layer is replaced by a Cluster-SVM classifier. This is used to speed up the process of classification.	The technique outperforms the state-of-the-art results on the dataset taken into account, requiring less training time.

(*continued on next page*)

Table 1 (*continued*)

Reference	Feature Extraction	Classification	Comments
Ding et al. (2018)	Pretrained AlexNet deep convolutional neural network model was used for feature extraction.	The features extracted with AlexNet and spectral features are used as the input for an SVM-based classifier.	The authors show that the deep convolutional neural networks can extract more accurate remote sensing image features, and significantly improve the overall accuracy of classification with respect to traditional feaure extraction methods.
Alaslani and Elrefaei (2018)	Pretrained AlexNet deep convolutional neural network model was used for feature extraction.	The features extracted with AlexNet are used as the input for an SVM-based classifier.	The authors focus their analysis on the suitability of CNN as a feature extractor. They show that features extracted with CNN are useful for improving the classification accuracy with respect to other techniques.
Minaee et al. (2017)	Pretrained VGGNet deep convolutional neural network model was used for feature extraction.	The features extracted with VGGNet are used as the input for an SVM-based classifier.	By using VGGNet as a feature extractor, authors show that it is possible to improve (in terms of classification accuracy) the results obtained by using other feature extraction techniques.
Bodapati and Veeranjaneyulu (2019)	Traditional CNN with three convolutional layers.	The features extracted with the CNN are used as the input for an SVM-based classifier.	The authors show that on the small dataset under examination, the use of a SVM outperforms the classification accuracy that can be obtained by a fully-connected neural network. These findings are difficult to generalize given that only one single dataset and one architecture were considered.
Thomaz et al. (2017)	Traditional CNN.	Multilayer perceptron trained with the features extracted from the CNN.	This work uses the same idea proposed in our paper. However, as pointed out by the authors, the results obtained are not robust due to the very small amount of data and the memory constraints they faced. Thus, they highlight the need for a deeper analysis that also considers different classifiers.
Corchs et al. (2017)	Pretrained AlexNet deep convolutional neural network model was used for feature extraction.	Ensemble model of five state-of-the-art classifiers: Naive Bayes, Bayesian Network, Nearest Neighbor, Decision Tree, and Linear Support Vector Machine.	The authors show that a deep pretrained network is able to extract features that increase the classification accuracy with respect to other feature extraction methods. Additionally, ensemble models, built by considering weak learners trained on these features, improve the classification accuracy with respect to single independent models.
Srivastava et al. (2017)	The authors extracted features with three pretrained deep convolutional neural networks (AlexNet, VGGNet, GoogleNet).	Ensemble model of SVM-based classifiers.	The authors show the suitability of ensemble models for combining the representation capability of features from deep networks with information captured from local features.
Hertel et al. (2017)	Traditional CNN architecture with five convolutional layers.	Classification performed by means of traditional fully-connected layers.	The authors show that convolutional networks are able to learn generic feature extractors that can be used for different tasks. This is an interesting study for the area of transfer learning.
Ayinde et al. (2019)	Modified version of the VGG architecture, where each layer of convolution is followed by a normalization layer.	Classification performed by means of traditional fully-connected layers.	This is an interesting study for better understanding of the features extracted by a CNN. In particular, the authors show how size, choice of activation function, and weight initialization impact redundant feature extraction of deep neural network models. The results achieved on well-known dataset show that the wider and deeper a network becomes, the higher is its tendency to extract redundant features.
Zhang et al. (2019)	1D-convolution capsule network separately extracts spatial and spectral information on spatial and spectral domains.	Classification Capsule Zhang et al. (2019).	The authors show that the proposed 1D-Convolution Capsule Network is superior to state-of-the-art methods with respect to both the accuracy and training effort.

design choices for the neural networks. The trained CNNs are then cut in such a way that the fully connected layers are removed, and only the flattened outputs from the convolutional layers remain. A high-level image feature in this context describes the output from the image filters from the last convolutional layer. The intermediate dataset is then generated as the flattened output of the trained convolutional layers along with the correct classification of the input image. The networks were created with the intention of producing suitable image filters, but the hyper-parameters were only fine-tuned to a reasonable degree, as this is not the focus of this paper. Although a high initial benchmark score in the first step is desirable to assure that the produced image filters are useful to the neural network, overall performance in the first step is not highly relevant to assess the performance in the second step because every classification algorithm gets the same input from the first step. Therefore, it is still a fair competition between the models' performance in the second step. A visualisation of the first step as derived from Fig. A.1 can be found in Fig. A.2.

The second script trains different classification algorithms on the intermediate datasets to test whether they can perform better than the original convolutional neural network. Section 3.3

describes the applied classification algorithms and associated hyper-parameters.

The quality of each classification algorithm is evaluated using top-*n* accuracy. Mainly top-1 and top-5 accuracy are used in the evaluation of the models, but more numbers are disclosed. Apart from that, the complexity of the models is measured in terms of the size of the models produced. For the MLP models, the size is determined as the size of the model structure in JSON format and the trained weights serialised in HDF5 format. The other models are serialised in Python Pickle format. The time complexity of training the individual models is not measured due to different training environments and conditions. Regardless, it is important to point out that the overall approach proposed in this paper is inevitably longer than simply training a convolutional neural network.

A visualisation of the second step as derived from the example in Fig. A.1 can be found in Fig. A.3.

To get robust and convincing results, several measures have been put into place. The approach is tested on different datasets. This will evaluate whether there are patterns or specific characteristics in the datasets that make certain algorithms perform better than others. These datasets have different sizes and different numbers of classes.

All algorithms' hyper-parameters are explored through a grid search with previously chosen options as outlined in Section 3.3. The training set is split from the test set, with a 5:1 split for CIFAR-10 and CIFAR-100[6] and a 3:1 split for ILSVRC-2012 (75 % training, 25 % test). This is done even before training the original CNN in the first step. The whole CNN then learns based on the data from the training set. The intermediate data for both the training and test sets is produced from that trained CNN model.

In the second step, the intermediate data created from the training set is used to train the classification algorithms and fine-tune the hyper-parameters. Hence, the same entries are used to train the models in the first and second steps.

Next, the training data is cross-validated with k-fold cross-validation with a value of $k = 10$. The validation data in this second step is part of the training data from the first step. The results of the cross-validations are saved, and *p*-values are calculated to assess statistical significance (see Appendix). The final algorithms' performance is validated on the initial test data that the algorithm did not previously used for training purposes. This procedure allows for a fair comparability of the results and ensures statistical validity.

3.2. Model architecture benchmarks

Different convolutional neural network architectures are used for each dataset. While the same network architectures can be applied to CIFAR-10 and CIFAR-100, many architectures that were designed for ILSVRC-2012 are not suitable for the much smaller images in CIFAR-10 and CIFAR-100. Through a pooling operation, the images are downsampled in the network. If too few pixels exist in the input layer, larger networks that use many pooling operations would downsample the image more than possible. Therefore, these network architectures would need to be modified in order to apply them to the CIFAR datasets.

3.2.1. ILSVRC-2012 architecture benchmarks

For the ILSVRC-2012 dataset, pre-existing models designed for the ImageNet dataset are leveraged to create the intermediate data. These networks are already pre-trained and thus do not require the computational effort of further training. Moreover, these models have been developed explicitly as benchmarks for the ImageNet dataset and should therefore perform well on these datasets. Three models were chosen because they achieve state-of-the-art results on the ImageNet dataset and are available with pre-trained weights in software suites.[7] The following models are used: Inception V3 (Szegedy et al., 2016), Xception (Chollet, 2016) and Inception ResNet V2 (Szegedy, Ioffe, Vanhoucke, & Alemi, 2017).

3.2.2. CIFAR architecture benchmarks

For the CIFAR-10 and CIFAR-100 datasets, identical model architectures are used, because the two datasets have the same input shape of $32 \times 32 \times 3$. Only the last fully connected output layer differs since CIFAR-10 expects 10 output neurons and CIFAR-100 has 100 output classes. All custom MLPs are designed with constant padding at the borders. For padding, the overlapping entries are filled with 0s for the convolutional layers and with $-\infty$ for the max-pooling layers.

The following MLP architectures were chosen for both CIFAR-10 and CIFAR-100. All networks were trained for 80 iterations.

- **CNN-1:** The first CNN's architecture is a 10-layer convolutional neural network[8] with a final fully connected output layer. The network has six convolutional layers which all use convolutional windows of $k_w = k_h = 3$ and a horizontal and vertical stride of 1. The first two convolutional layers have depths of 128, the following two convolutional layers have depths of 256 and the last two convolutional layers have depths of 512. After each convolutional layer, an exponential linear unit (ELU) activation is applied. After the activation of every second convolutional layer, a pooling layer with a pooling window of 2×2 using max-pooling is applied. The horizontal and vertical strides are also 2, such that the image width and height are approximately halved[9] after each pooling operation. The pooling operations do not use padding at the borders.[10] After every pooling operation, a dropout layer is added. The dropout probability increases from 0.1 after the first pooling layer to 0.25 after the second one and 0.5 after the third pooling layer. After this, the output is flattened, and two dense layers follow. The first dense layer has 1024 neurons and is followed by an ELU activation and a dropout layer with a probability of 0.5. The second dense layer (output layer) makes the mapping to the number of output classes, 10 or 100. This final output is compressed to between 0 and 1 through the application of a softmax function. This network produces 2048 high-level intermediate features after its convolutional layers.
- **CNN-2:** The second convolutional neural network is constructed to be smaller than the first network and only has seven hidden layers. It follows the same design principle as the CNN-1 of creating two convolutional layers followed by one max-pooling layer. The activation functions after the convolutional layers are changed to rectified linear units (ReLU). This construct is only used twice instead of three times, as in the first convolutional neural network. This means only four convolutional and two pooling layers are used in total. The parameters for the pooling layers are left the same. The depths of the convolutional layers are reduced to 32 for the first block (first two convolutional

[6] Per default dataset options.

[7] In this work we used the models provided by Keras (www.keras.io).

[8] only counting hidden convolutional, dense and pooling layers.

[9] approximately halved depending on whether the width and height are even or uneven numbers.

[10] Since the input images for both CIFAR datasets have 32 pixels in both dimensions, they can be perfectly divided by 2 five times. The width and height stay even during these divisions, so border handling is not necessary.

layers) and 64 for the latter block (last two convolutional layers). Dropout after these blocks is kept constant at 0.25. The flattened output is then passed to a dense layer with 512 neurons, followed by a ReLU activation and a dropout layer with a probability of 0.5. The output layer is identical in structure to that of CNN-1, making the squashed prediction of 10 or 100 classes with a fully connected layer. This network structure produces 2304 high-level intermediate features as input to the fully connected layers.

- **SimpleNet:** SimpleNet has 18 hidden layers, of which 13 are convolutional layers and five are max-pooling layers. The output layer making the final prediction is a regular fully connected layer. The convolutional layers use 3×3 convolutional windows for the first 10 and the 13th layers, while the layers 11 and 12 use 1×1 convolutions. The depth of the convolutional layers increases from 64 in the first layer to 128 for layers 2 through 6, 256 for layers 7 to 9, 512 for layer 10, and 2048 for the 11th convolutional layer. The convolutional layers 12 and 13 have a depth of $k_d = 256$. All convolutional layers have horizontal and vertical strides of 1 and apply padding at the borders such that the image dimensions are not altered after the convolutional operation. Batch normalisation and ReLU activation are applied after convolutional layers 1 to 6 and 8 to 10. In layer 7, batch normalisation and the activation function are only applied after the pooling operation. Convolutional layers 11 to 13 only use a ReLU activation after the convolutional layer, without batch normalisation. Dropout is consistently applied after batch normalisation, activation, or max-pooling (where applicable) for convolutional layers 1 through 8 as well as 10 and 11, with a probability of 20%. In the ninth layer, the dropout function was applied before the pooling operation. The five max pooling operations were applied after convolutional layers 4, 7, 9, 12, and 13. The pooling operations all use pooling windows of 2×2 with horizontal and vertical strides of 2 and no padding at the borders. All convolutional layers are initialised with the Glorot normal initialiser (Glorot & Bengio, 2010). SimpleNet produces 256 high-level image features.
- **VGG-19:** VGG-Net won the localisation task competition on ImageNet in 2014 and placed second in the classification task after GoogLeNet. The general idea is to build deeper networks with smaller convolutional filters that essentially cover the same receptive fields but enable more transformations and non-linearities (Simonyan & Zisserman, 2014). The VGG network of this paper is a version adapted to the CIFAR datasets with 19 hidden layers (VGG-19). Of the 19 hidden layers, 13 layers are convolutional, five layers are max-pooling layers, and there is one dense layer at the end before the fully connected output layer. As previously mentioned, the convolutional layers all use 3×3 image filters with padding at the borders and with horizontal and vertical strides of 1, such that the image dimensions are maintained. As the network gets deeper, the depth of the convolutional layers increases as well. The first two convolutional layers have depths of 64, the proceeding two layers have depths of 128, the subsequent three layers have depths of 256, and the following six convolutional layers have depths of 512. The final dense hidden layer has 512 neurons. After each convolutional layer, a ReLU activation function is applied. The same goes for the final hidden fully connected layer. After the activation, batch normalisation is applied. The five max-pooling layers are layers 3, 6, 10, 14, and 18. They all use a pooling window of size 2×2 with horizontal and vertical strides of 2. Over the whole network, dropout is only applied in some layers. Nonetheless, the principal of increasing dropout probabilities is adhered to. Throughout the network, dropout is applied after layers 1, 4, 7, 8, 11, 12, 15, 16, 18, and 19. Layer 1 has a dropout probability of 0.3, and layers 18 and 19 have a dropout probability of 0.5. The layers between layers 2 and 18 use a dropout probability of 0.4. VGG-19 produces 512 high-level image features as input to the subsequent classifier.

3.3. Classification algorithms used

After creating the intermediate dataset, several classification algorithms are applied to the intermediate dataset. The following classification algorithms are used with the specified parameters exhaustively tested in a grid search to enable an objective comparison:

- **Support vector machine (SVM)**: 12 different SVM configurations are tested in the grid search. The first four SVMs use a linear kernel with C[11] values of 1, 10, 100 and 1000. The other eight SVM configurations use RBF as the kernel, with C values of 1,10,100 and 1000 and gamma (γ)[12] values of 1×10^{-3} and 1×10^{-4}.
- **Logistic regression (LR)**: 12 LR configurations are tested. For the loss function, each model either uses an $\ell1$ or $\ell2$ loss as the penalty of the model, while the value for c is set in the range of $1 \times 10^{-3}, 1 \times 10^{-2}, \ldots, 1 \times 10^{2}$.
- **K-nearest neighbours (KNN)**: 6 KNN configurations are tested. KNN can get computationally very expensive, especially for large values of k. The benchmark was done with $k \in 1, 5, 10$ and the leaf size either being 1 or 5 as well. All configurations use Euclidean distance as the distance metric.
- **Random forests estimator (RFE)**: 6 different RFE configurations are tested. The number of estimators is set to either 10, 100, or 1000, combined with Gini or Entropy diversion measure. The estimators are decision trees.
- **AdaBoost classifier (ADB)**: AdaBoost is benchmarked on nine configurations. Models with either 10, 100, or 1000 estimators are combined with learning rates of 1.0, 0.5, or 0.1. The estimators are decision trees.
- **Gradient boosting classifier (GBC)**: Gradient Boosting's benchmark parameter combinations total to six models. The models have 10 or 100 estimators, with learning rates of 0.05, 0.1 or 0.5. The estimators are decision trees.
- **XGBoosting classifier (XGB)**: XGBoosting is benchmarked on 16 configurations. The number of estimators is either 10 or 100. The learning rate is set to be 0.1 or 0.01. The estimators are decision trees. The maximum depth of each weak tree learner is limited to 1, 3, 5, or 10.

These classification algorithms are further compared against various multi-layer perceptron architectures. All MLP architectures are trained with categorical cross-entropy loss, an Adam optimiser with learning rates of 0.0001 and 0.001, and with accuracy as the optimisation metric. The models are trained for 10,20,30,50 and 100 epochs.[13] The final layer is encoded with one neuron per output class.

Regarding topology, we selected three architectures that were inspired by the fully connected layers of existing convolutional neural network architectures. These architectures were fine-tuned

[11] C is the parameter associated with the slack variables from ξ.

[12] γ is the free parameter of the RBF kernel function.

[13] Although this could be done more efficiently, the models are created from scratch every time to leverage the differences of random initialisation. Else, it would be sufficient to create a model for the maximum number of iteration and save the model after each indicated number of iteration

after a preliminary process in which several architectures were tested.

- **MLP-1**: MLP-1 is a neural network with one fully connected hidden layer and one output layer. Its hidden layer is a dense layer with 1024 neurons, followed by an ELU activation function and a dropout layer with a probability of 0.5. The output layer, with as many neurons as final classes, is followed by a softmax activation to squash the values to between 0 and 1.
- **MLP-2**: MLP-2 has one fully connected hidden and one output layer. Its hidden dense layer has 512 neurons, followed by a ReLU activation function and a dropout layer with a probability of 0.5. The dense output layer (with as many neurons as final classes) is squashed to between 0 and 1 with a softmax activation function.
- **MLP-3**: MLP-3 only has one fully connected output layer. Hence, it directly performs the mapping from the flattened image features to the output. The only layer has as many neurons as final classes, followed by a softmax activation function. Technically, this is not an MLP since it consists only of an input and an output layer.

Since these three architectures are combined with two possible learning rates and five different epochs, the total number of created models is 30. During training, all model architectures converged (thus suggesting that the hyper parameters selected with grid-search have suitable values).

4. Results

This section summarises and interprets the results that were achieved using the experimental setup outlined in Section 3. The experimental setup consists of an initial step of creating candidate intermediate datasets through different CNNs, followed by a benchmark of different classifiers on these intermediate datasets. First, the results and observations from the intermediate data creation step are displayed in Section 4.1. After that, the results from the second classification step on these intermediate datasets are presented and described in Sections 4.2 and 4.3. The achieved results are then further analysed and interpreted in Section 4.4.[14]

4.1. Output from convolutional neural network structures

The following section discusses the main findings from the creation of the intermediate datasets through the training of CNNs on the initial datasets.

The networks were trained for 80 iterations, and they all seem to have converged for both the training and the test datasets. As described in Section 3.2.2, the algorithms' architectures were selected after trying out different architectures and hyper-parameters, which is why the set of CNN models does not contain any entirely erroneously trained models.

4.1.1. CIFAR-10 intermediate observations

For CIFAR-10, the results can be found in Table 2. VGG-19 is not trained separately because it is available as a pre-trained model for CIFAR-10.[15]

From the results reported in Table 2, it is notable that VGG-19 has the best final accuracy. From the manually trained models, SimpleNet reaches the highest testing and training accuracy values. It also converges the fastest on both training and test data. Nonetheless, the CNN-1 architecture achieves only marginally worse results, with SimpleNet outperforming the test accuracy of CNN-1 by 3.5 percentage points. CNN-2 performs worse than the aforementioned two architectures, which could be caused by the reduced complexity of the model, making CNN-2 unable to observe more complex patterns. Overall, there are some indications of overfitting of the algorithms on the training data compared to the test data. Nonetheless, this does not worsen the test datasets' accuracy scores. Since the test accuracy did not decrease, but stayed steady or even improved slightly, the networks are considered to be well suited for further analyses. Compared to the results from the pretrained VGG-19 network, the three benchmark models performed slightly worse (between 5 to 12 percentage points) on unseen data.[16] From the size of the VGG-19 network, it can be seen that it is more complex than the other networks, which is likely why it can observe more complex patterns in the image data.

4.1.2. CIFAR-100 intermediate observations

The results for CIFAR-100 can be found in Table 3. As for CIFAR-10, a pre-trained VGG-19 model is used.[17]

To summarise the results of Table 3, CNN-1 and SimpleNet achieve a similar accuracy on their training data (around 0.93). Nonetheless, SimpleNet can learn more quickly and generalises better over unseen data, and its test data accuracy is about 2 percentage points higher. As on CIFAR-10, CNN-2 performs worse than the other architectures, probably because it can capture fewer non-linear relationships. The results from the custom-trained neural networks are worse than the ones achieved with the pre-trained and more complex VGG-19. VGG-19 seems to detect more patterns in the training data and produces good results over unseen data,[18] reaching an accuracy of 0.7048. This is almost 10 percentage points higher than the second best from SimpleNet. Apart from the increased complexity of VGG-19, another reason for its superiority could lie within the data preprocessing stage.

4.1.3. ILSVRC-2012 subset intermediate observations

An overview of the model specifications created from the benchmarked CNN architectures on the subset of ILSVRC-2012 with 100 classes can be found in Table 4. Compared to the CIFAR datasets, the intermediate datasets from the ILSVRC-2012 CNN networks contain many more high-level image features (intermediate features).

The best performing model on both top-1 and top-5 is Inception ResNet V2 with a top-1 accuracy of 0.8030 and a top-5 accuracy of 0.9530. In terms of size, it is also the biggest and most complex model. Apart from the structural advantages that the Inception ResNet architecture brings, the additional model complexity enables the model to learn more complex non-linear relationships within the data. Contrarily to the model's size, the produced training and test data sizes for the subsequent step are the smallest, with 1620 MB for training and 539 MB for the test data. This indicates that the network's complexity does not substantially lie in its last fully connected layers. The number of produced features

[14] CIFAR-10, CIFAR-100, and the used subset of ILSVRC-2012 are referred to as initial or original datasets. The CNNs trained or obtained on these initial datasets (e.g., CNN-1) are called original, trained, or benchmark CNNs. The produced datasets from the CNNs are referred to as intermediate datasets, usually denoted with the corresponding CNN and original dataset (e.g., "CNN-1 intermediate dataset trained on CIFAR-10"). This information is omitted when it can be inferred from context. The benchmarked classification algorithms are named benchmark classifiers or, generally, classification algorithms. When top-*n* accuracy is mentioned, we refer refer to the top-*n* accuracy on test data, unless explicitly stated differently.

[15] Pre-trained model from https://github.com/geifmany/cifar-vgg.

[16] VGG-19 was not trained by the author thus it is not possible to determine which data points were used as training and test data. A comparison between the final model's performance and the performance of the other models on unseen data is therefore not completely accurate.

[17] Pre-trained model from https://github.com/geifmany/cifar-vgg.

[18] VGG-19 was not trained by the author; thus, it is not possible to determine which data points were used as train and test data. A comparison between the final model's performance and the performance of the other models on unseen data is therefore not completely accurate.

Table 2
CIFAR-10 intermediate dataset results.

Architecture	Top-1 test accuracy	Top-1 train accuracy	Model size [MB]	Train data size [MB]	Test data size [MB]	lIntermediate features	Param
CNN-1	0.8504	0.9846	26.5	636.8	127.7	2048	6.7M
CNN-2	0.8113	0.9155	5	520.2	104.4	2304	1.3M
SimpleNet	0.8852	0.9869	22.1	69.6	14.3	256	5.5M
VGG-19	0.9359	0.9359	60.1	143.2	28.9	512	15M

Table 3
CIFAR-100 intermediate dataset results.

Architecture	Top-1 test accuracy	Top-1 training accuracy	Model size [MB]	Training data size [MB]	Test data size [MB]	Intermediate features	Param
CNN-1	0.5834	0.9297	26.5	602.5	120.5	2048	6.7M
CNN-2	0.4750	0.6308	5	501.9	100.4	2304	1.3M
SimpleNet	0.6099	0.9303	22.1	72.0	14.4	256	5.5M
VGG-19	0.7048	0.7048	60.3	141.3	28.3	512	15M

Table 4
ILSVRC-2012 intermediate dataset results for 100 classes.

Architecture	Top-1 accuracy	Top-5 accuracy	Model size [MB]	Training data size [MB]	Test data size [MB]	Intermediate features	Param
Inception V3	0.7790	0.9370	92	2200	732	131,072	21.8M
Xception	0.7900	0.9450	88	3240	1080	204,800	20.8M
Inception ResNet	0.8030	0.9530	215	1620	539	98,304	54.3M

from the flattened convolutional layer is also the lowest among the compared models. Inception ResNet V2 produces 98,304 features for the intermediate dataset. Inception V3 and Xception seem to be similar in overall model complexity with Inception V3 having 21.8M parameters while Xception only has 20.8M parameters. Nonetheless, Xception can outperform Inception on both top-1 and top-5 test accuracy. Xception reaches a top-1 test accuracy of 0.7900, while Inception V3 only reaches a top-1 test accuracy of 0.7790. The top-5 test accuracy of Xception is 0.9450 while the top-5 test accuracy of Inception V3 is 0.9370. Regarding the complexity of the produced intermediate datasets, Xception generates the intermediate dataset with the most features: 204,800. Inception V3's intermediate dataset has 131,072 features. The train and test intermediate data sizes for Inception V3 are 2200 MB and 732 MB respectively. For Xception, the training intermediate dataset size is 3240 MB, and its test set has a size of 1080 MB.

4.2. MLP architecture comparison on intermediate datasets

This section describes the main findings from retraining MLP architectures on the intermediate datasets. This benchmark mainly shows how well a separated learning process can imitate the original learning process of a full CNN. Because the intermediate data is created by removing the fully connected layers from the trained CNNs, training a fully connected neural network on this intermediate dataset should be able to achieve at least comparable results to the initially trained models, given that the network structure used in the second step is similar to the one that is used for the final classification in the first step. The observations in this section are based on the intermediate datasets created from CIFAR-100. Nonetheless, the findings generalise to other datasets because the observations on CIFAR-100 are similar to the ones on the other benchmarked datasets.

As described in Section 3.3, the MLP architectures are trained on the intermediate output data for 10, 20 and 50 epochs combined with learning rates of 0.001 and 0.0001. Because this is done with 10-fold cross-validation, this leads to 60 neural networks being created per architecture with a total of 1600 epochs of training 45,000 entities per epoch. Nonetheless, it can be observed that some of these hyper-parameters do not work well on the intermediate dataset. To exemplify this, the results on the first intermediate dataset, created through CNN-1, can be found in Appendix H. Most networks already converge after 10 epochs, and further training does not lead to any improvements, neither on the training nor on the validation set (during cross-validation). Fig. C.1 shows the behaviour of MLP-1 being trained on CNN-1 intermediate data after 10 epochs, whereas Fig. C.2 shows the same training procedure after 50 epochs. It can be seen that the networks seemingly cannot detect any new patterns after 10 iterations. As for the learning rate, a lower learning rate works better than a higher learning rate. Both on the training and the validation set, a higher learning rate leads to less smooth learning curves with the networks making rather big jumps without converging to an optimal value. This can be seen by comparing Fig. C.2 (illustrating MLP-1 being trained on CNN-1 intermediate data for 50 epochs with a learning rate of 0.0001) to Fig. C.3 (illustrating the same network being trained with a higher learning rate of 0.001). The same behaviour is observable for the other MLP architectures. Experimentation with an even lower learning rate than 0.0001 did not lead to any significant improvements.

For the three MLP architectures, the results in Tables 5–8 were achieved on the intermediate datasets from CNN-1, CNN-2, SimpleNet and VGG-19 respectively. Tables E.1, E.2, E.3 and E.4 in Appendix E show the p-value of a two-sided student's t-test for the respective validation results. From the p-values, the statistical significance levels of the results can be derived.

A comparison of the models from Tables 5–8, show that the test accuracy scores of some intermediate datasets are consistently lower than others. For example, the accuracy scores on CNN-2 intermediate data in Table 6 are between 0.3687 (MLP-3, LR-0.001) and 0.4498 (MLP-1, LR-0.0001) on the test dataset, while the accuracy scores on VGG-19 intermediate data in Table 8 are between 0.6927 (MLP-1, LR-0.001) and 0.7080 (MLP-2, LR-0.0001) on the test dataset. The VGG-19 network already performed much better in the previous step of creating the intermediate dataset (see Table 3). This emphasises the importance of the previously learned image features for the overall model performance in this subsequent step.

Concerning MLP architectures, it is hard to pick a configuration that works best for all datasets. Of the six compared configurations (3 MLP architectures, 2 learning rates), MLP-1 with a learning rate of 0.0001 performs the best on CNN-2 and

Table 5
CIFAR-100 MLP classification results on CNN-1 intermediate data, where LR-1 is a learning rate of 0.0001 and LR-2 is a learning rate of 0.001.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Top-10 test accuracy	Top-20 test accuracy	Average validation accuracy	Average training Accuracy	Model size
MLP-1 LR-1	0.5585	0.6770	0.8117	0.8790	0.9349	0.9896	0.9997	8611KB
MLP-2 LR-1	0.5630	0.6860	0.8148	0.8874	0.9422	0.9884	0.9997	4313KB
MLP-3 LR-1	0.5592	0.6753	0.8002	0.8717	0.9310	0.9888	0.9998	811KB
MLP-1 LR-2	0.5415	0.5976	0.6277	0.6521	0.6922	0.9680	0.9929	8161KB
MLP-2 LR-2	0.5500	0.6381	0.7006	0.7305	0.7645	0.9738	0.9979	4313KB
MLP-3 LR-2	0.5293	0.6466	0.7803	0.8593	0.9288	0.9681	0.9979	811KB

Table 6
CIFAR-100 MLP classification results on CNN-2 intermediate data, where LR-1 is a learning rate of 0.0001 and LR-2 is a learning rate of 0.001.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Top-10 test accuracy	Top-20 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP-1 LR-1	0.4498	0.5773	0.7272	0.8279	0.9060	0.4494	0.9069	9635KB
MLP-2 LR-1	0.4402	0.5692	0.7311	0.8235	0.9058	0.4402	0.8191	4825KB
MLP-3 LR-1	0.4223	0.5439	0.7051	0.8144	0.9055	0.4235	0.7330	911KB
MLP-1 LR-2	0.4017	0.5247	0.6844	0.7843	0.8807	0.4102	0.8852	9635KB
MLP-2 LR-2	0.3741	0.4974	0.6616	0.7748	0.8740	0.3762	0.6694	4825KB
MLP-3 LR-2	0.3687	0.4911	0.6520	0.7653	0.8717	0.3713	0.8906	911KB

Table 7
CIFAR-100 MLP classification results on SimpleNet intermediate data, where LR-1 is a learning rate of 0.0001 and LR-2 is a learning rate of 0.001.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Top-10 test accuracy	Top-20 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP-1 LR-1	0.6326	0.7549	0.8744	0.9320	0.9683	0.9838	0.9970	1443KB
MLP-2 LR-1	0.6265	0.7531	0.8777	0.9354	0.9715	0.9589	0.9825	729KB
MLP-3 LR-1	0.6067	0.7394	0.8642	0.9276	0.9699	0.9417	0.9694	111KB
MLP-1 LR-2	0.6066	0.7277	0.8549	0.9172	0.9638	0.9430	0.9711	1443KB
MLP-2 LR-2	0.6027	0.7316	0.8601	0.9220	0.9653	0.9275	0.9643	729KB
MLP-3 LR-2	0.6137	0.7410	0.8602	0.9207	0.9593	0.9557	0.9707	111KB

Table 8
CIFAR-100 MLP classification results on VGG-19 intermediate data, where LR-1 is a learning rate of 0.0001 and LR-2 is a learning rate of 0.001.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Top-10 test accuracy	Top-20 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP-1 LR-1	0.7048	0.7960	0.8646	0.9069	0.9423	0.9962	0.9980	2467KB
MLP-2 LR-1	0.7080	0.8051	0.8817	0.9193	0.9533	0.9968	0.9984	1241KB
MLP-3 LR-1	0.7052	0.8034	0.8782	0.9183	0.9525	0.9969	0.9982	211KB
MLP-1 LR-2	0.6927	0.7574	0.8059	0.8312	0.8583	0.9940	0.9957	2467KB
MLP-2 LR-2	0.6994	0.7767	0.8387	0.8798	0.9150	0.9955	0.9966	1241KB
MLP-3 LR-2	0.6973	0.7829	0.8498	0.8938	0.9317	0.9948	0.9978	211KB

SimpleNet intermediate data in terms of top-1 test accuracy. For the other two intermediate datasets, MLP-2 achieves higher accuracy scores.

To sum up, this section shows that the separation of the learning process to create image features and train fully connected neural networks on those image features performs comparably to the initial training process of an entire CNN. Although the final top-1 test accuracy was slightly lower on CNN-1 and CNN-2 intermediate data than on the initially trained CNN-1 and CNN-2 models, the contrary was the case for the VGG-19 and SimpleNet intermediate data. Therefore, it can be concluded that a detached final classifier is in general still able to learn as well as the full network in practice. These observations were also confirmed during the other training processes on CIFAR-10 and the ILSVRC-2012 subset.

4.3. Comparing different classification algorithms on image features

The following section lists the accuracy values that were achieved in the second step on the intermediate datasets produced on CIFAR-10, CIFAR-100, and ILSVRC-2012. The tables are split by the intermediate datasets. Each table presents a comparison of different classifiers trained on the respective intermediate datasets. Because four CNNs were used on the CIFAR networks to create the intermediate datasets and three CNNs were used on ILSVRC-2012, the number of intermediate datasets benchmarked in this section totals 11. Each section contains a textual description of the findings, pointing out the key points from the tables to focus on and comparing the performance of the different classifiers. Apart from the top-*n* test accuracies, the tables list the average validation and average training accuracy achieved over the 10-fold cross-validation. Moreover, the model size of the best performing parameter selection that achieved these results is included in each row. The best performing model parameters can be found in Appendix G and Appendix H.

4.3.1. CIFAR-10 results

The performance results of the classification algorithms on the intermediate datasets produced from CNN-1, CNN-2, SimpleNet and VGG-19 on CIFAR-10 can be found in Tables 9–12 respectively. The corresponding model configurations for the best performing models of each architecture can be found in Tables G.1, G.2, G.3 and G.4 in Appendix G. The significance matrices on the validation data benchmarks are presented in Tables D.5, D.6, D.7 and D.8.

To summarise the results reported in Tables 9–12, the models of most intermediate datasets were not able to consistently produce better values than the initially benchmarked CNN models. Only for SimpleNet intermediate data, six of the eight benchmarked models score better than the SimpleNet benchmark score. For most intermediate datasets, the best top-1 accuracy values are

Table 9
CIFAR-10 final classification results on CNN-1 intermediate data.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP	0.8490	0.9367	0.9880	0.9987	0.9994	4200KB
SVM	0.8454	0.9321	0.9850	0.9819	1.0000	497,200KB
LR	0.8469	0.9339	0.9868	0.9998	0.8130	191KB
KNN	0.7369	0.8503	0.9385	0.8245	0.8691	1,300,000KB
RFE	0.8120	0.9202	0.9893	0.9254	1.0000	1,990,000KB
ADB	0.7967	0.9086	0.9833	0.9119	0.9614	843KB
GBC	0.8175	0.9204	0.9874	0.9722	1.0000	1200KB
XGB	0.8182	0.9198	0.9877	0.9527	0.9985	2400KB

Table 10
CIFAR-10 final classification results on CNN-2 intermediate data.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP	0.7686	0.8974	0.9851	0.7908	0.8893	4800KB
SVM	0.7952	0.9140	0.9869	0.7851	1.0000	724,600KB
LR	0.7622	0.8933	0.9810	0.7750	0.8130	191KB
KNN	0.6170	0.6577	0.8050	0.5789	1.0000	1,460,000KB
RFE	0.6994	0.8496	0.9736	0.6962	1.0000	206,300KB
ADB	0.6446	0.8306	0.9699	0.6456	0.6616	844KB
GBC	0.7022	0.8539	0.9735	0.7015	0.7818	1300KB
XGB	0.7353	0.8788	0.9790	0.7331	1.0000	22,100KB

Table 11
CIFAR-10 final classification results on SimpleNet intermediate data.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP	0.8941	0.9591	0.9929	0.9994	0.9997	1100KB
SVM	0.8945	0.9605	0.9921	0.9995	1.0000	2800KB
LR	0.8806	0.9572	0.9945	1.0000	1.0000	27KB
KNN	0.8960	0.9366	0.9682	0.9982	0.9987	162,900KB
RFE	0.8933	0.9632	0.9958	0.9986	1.0000	221,600KB
ADB	0.8766	0.9472	0.9916	0.9909	0.9950	843KB
GBC	0.8907	0.9594	0.9937	0.9982	1.0000	1300KB
XGB	0.8881	0.9586	0.9940	0.9971	1.0000	1500KB

Table 12
CIFAR-10 final classification results on VGG-19 intermediate data.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP	0.9352	0.9696	0.9835	0.9999	0.9999	1100KB
SVM	0.9349	0.9797	0.9952	0.9999	0.9999	3500KB
LR	0.9346	0.9772	0.9932	0.9999	0.9999	46KB
KNN	0.9350	0.9541	0.9757	0.9999	0.9999	324,800KB
RFE	0.9350	0.9661	0.9864	0.9999	1.0000	868KB
ADB	0.9330	0.9659	0.9927	0.9998	1.0000	844KB
GBC	0.9318	0.9523	0.9803	0.9996	1.0000	1100KB
XGB	0.9327	0.9683	0.9890	0.9998	1.0000	273KB

similar to the benchmarked values. Apart from the retrained MLP models, the models from SVMs and LR frequently appear among the best performing models. LR has the advantage of small model sizes. Although SVMs outperforms MLPs at times, the model sizes of SVMs have high variance. For example, the best performing model on SimpleNet intermediate data is an SVM model of 2800KB size, while the best performing model on the CNN-2 intermediate data is an SVM model of size 724,600KB. Considering model size and performance, MLPs and LR perform best. Regarding the tree-based models, ADB performs the worst among these. Overall, XGB manages to produce slightly better results than GBCs, particularly on CNN-2 intermediate data. Nonetheless, GBCs are remarkably constant in model size between 1100 and 1300KB, whereas XGB fluctuates more. Between RFEs and XGB, it is hard to choose whether the bagging or the boosting approach performs better overall. While RFEs reach slightly higher top-1 test accuracies on VGG-19, SimpleNet, and CNN-1 intermediate data, XGB performs much better on CNN-2 intermediate data. Nonetheless, all models produced from boosting algorithms are much smaller in size than RFEs. The KNN model exhibits some interesting behaviour: although it performs comparably to the best model on VGG-19 and outperforms all other models on SimpleNet, its performance on CNN-1 and CNN-2 is much lower than the other benchmarked models. While the CNN-2 intermediate data uses a model with only 1 neighbour, all other intermediate datasets use models with 10 neighbours. Apart from these observations, KNN generally produces huge models, sometimes even over 1GB in size, for example, on CNN-1 and CNN-2. For higher values of n, the top-*n* accuracy of KNN gets worse in comparison to the other models.

4.3.2. CIFAR-100 results

The results of the classification algorithms on the intermediate datasets from CNN-1, CNN-2, SimpleNet and VGG-19 on CIFAR-100 can be found in Tables 13 respectively. The corresponding model configurations for the best performing models of each architecture can be found in Tables H.1, H.2, H.3 and H.4 in Appendix H. The

Table 13
CIFAR-100 final classification results on CNN-1 intermediate data.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Top-10 test accuracy	Top-20 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP	0.5630	0.6860	0.8148	0.8874	0.9422	0.9884	0.9997	4313KB
SVM	0.6030	0.7250	0.8515	0.9146	0.9623	0.9820	0.9998	791,833KB
LR	0.5706	0.6797	0.7994	0.8705	0.9269	0.9935	0.9996	1605KB
KNN	0.3284	0.3320	0.3547	0.3929	0.4694	0.3769	0.9998	1,266,158KB
RFE	0.3192	0.4173	0.5485	0.6525	0.7514	0.3576	0.9998	3,223,027KB
ADB	0.2042	0.3058	0.4851	0.6325	0.7876	0.2482	0.2758	335KB
GBC	0.2547	0.3602	0.5422	0.6111	0.7493	0.4369	0.9994	12,150KB
XGB	0.3326	0.4397	0.5692	0.6801	0.8115	0.4721	0.9996	4100KB

Table 14
CIFAR-100 final classification results on CNN-2 intermediate data.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Top-10 test accuracy	Top-20 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP	0.4498	0.5733	0.7272	0.8279	0.9060	0.4494	0.9988	9635KB
SVM	0.4741	0.6020	0.7554	0.8499	0.9242	0.4576	0.9770	890,827KB
LR	0.4518	0.5702	0.7266	0.8281	0.9089	0.4417	0.6336	1805KB
KNN	0.2878	0.2912	0.3139	0.3538	0.4308	0.2666	0.9998	1,424,346KB
RFE	0.2931	0.3822	0.5211	0.6243	0.7298	0.2793	0.9998	3,518,103KB
ADB	0.1728	0.2682	0.4336	0.5793	0.7313	0.1732	0.1845	335KB
GBC	0.2880	0.3402	0.5041	0.6034	0.7021	0.3044	0.9555	12,150KB
XGB	0.3009	0.3798	0.5386	0.6403	0.7567	0.3518	0.9993	3400KB

Table 15
CIFAR-100 final classification results on SimpleNet intermediate data.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Top-10 test accuracy	Top-20 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP	0.6326	0.7549	0.8744	0.9320	0.9683	0.9838	0.9970	1443KB
SVM	0.6380	0.7615	0.8826	0.9419	0.9772	0.9801	0.9989	80,543KB
LR	0.6247	0.7439	0.8554	0.9141	0.9543	0.9937	0.9997	205KB
KNN	0.6066	0.7286	0.8237	0.8465	0.8646	0.8380	0.8904	158,842KB
RFE	0.5955	0.7216	0.8436	0.9029	0.9460	0.8258	0.9998	2,049,021KB
ADB	0.3850	0.5275	0.7259	0.8408	0.9261	0.5165	0.5377	335KB
GBC	0.5532	0.6802	0.7947	0.8834	0.9339	0.8812	0.9995	12,150KB
XGB	0.5894	0.7107	0.8473	0.9132	0.9450	0.9683	0.9996	2200KB

Table 16
CIFAR-100 final classification results on VGG-19 intermediate data.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Top-10 test accuracy	Top-20 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP	0.7080	0.8051	0.8817	0.9193	0.9533	0.9968	0.9984	1241KB
SVM	0.7114	0.8113	0.8950	0.9336	0.9651	0.9972	0.9978	32,203KB
LR	0.7071	0.8056	0.8768	0.9133	0.9568	0.9969	0.9983	406KB
KNN	0.7095	0.7644	0.7892	0.8016	0.8262	0.9970	0.9972	317,030KB
RFE	0.7095	0.8086	0.8886	0.9266	0.9568	0.9969	0.9999	2,397,625KB
ADB	0.6629	0.7505	0.8453	0.8939	0.9403	0.9854	0.9897	335KB
GBC	0.6747	0.7502	0.8147	0.8534	0.8939	0.9911	0.9999	12,150KB
XGB	0.6897	0.7707	0.8423	0.8851	0.9260	0.9948	0.9999	4787KB

significance matrices on the validation data benchmarks are presented in Tables E.5, E.6, E.7 and E.8.

To summarise the results reported in Tables 13–16, the best performing model on all intermediate datasets from CIFAR-100 is an SVM model. After that, LR and MLPs follow, with LR being superior on CNN-1 and CNN-2 intermediate data and MLPs performing better on SimpleNet and VGG-19 intermediate data. SVMs also outperform the top-1 accuracy of the CNN benchmarks from the first step (Table 3) on CNN-1, SimpleNet, and VGG-19 intermediate data. On SimpleNet and VGG-19 intermediate data, the retrained MLP and LR models are also superior in performance to the initial benchmark. Considering the external benchmarks on CIFAR-100 from Table B.2, the achieved top-1 test accuracy scores on VGG-19 intermediate data from Table 16 rank among the best available model benchmarks on CIFAR-100.

The bagging and boosting approaches perform similarly concerning top-1 accuracy, but their performance is generally worse than the other benchmarked algorithms. Similar performance to the other classification algorithms is only achieved by RFEs on VGG-19 intermediate data. Among the boosting algorithms, XGB is the best algorithm on all intermediate datasets. ADB has the lowest top-1 test accuracy among all classifiers on every intermediate dataset. Regarding model size, RFEs produce much larger models than all other models, with the largest model being 3.5GB in size from CNN-2 intermediate data. The boosting algorithms generally produce rather small models. The smallest in each intermediate dataset benchmark is either ADB or LR. The latter additionally has good performance values despite being very small in size, up to seven times smaller than MLPs (see Table 15). The top-5 accuracy scores generally follow the same trend as the top-1 accuracy

Table 17
ILSVRC-2012 final classification results on inception V3 intermediate data.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Top-10 test accuracy	Top-20 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP	0.7936	0.8984	0.9480	0.9672	0.9808	0.7760	0.8920	537,300KB
SVM	0.3552	0.6184	0.6576	0.7000	0.7576	0.3168	0.9997	1,204,000KB
LR	0.8512	0.9304	0.9768	0.9920	0.9944	0.8459	0.9997	104,900KB
KNN	0.7608	0.7680	0.7792	0.7904	0.8064	0.7509	0.9997	4,580,000KB
RFE	0.8256	0.9176	0.9656	0.9792	0.9880	0.8157	0.9997	1,780,000KB
ADB	0.2024	0.2856	0.4360	0.5560	0.7048	0.1341	0.3835	3800KB
GBC	0.4152	0.5488	0.6912	0.7448	0.7816	0.4515	0.9997	6100KB
XGB	0.7888	0.8848	0.9336	0.9560	0.9752	0.7461	0.9973	4700KB

Table 18
ILSVRC-2012 final classification results on Xception intermediate data.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Top-10 test accuracy	Top-20 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP	0.8208	0.9128	0.9488	0.9664	0.9800	0.8245	0.9136	419,700KB
SVM	0.7608	0.8800	0.9216	0.9512	0.9696	0.7483	0.9997	1,530,000KB
LR	0.8528	0.9336	0.9792	0.9936	0.9976	0.8528	0.9976	163,800KB
KNN	0.8296	0.9152	0.9464	0.9552	0.9592	0.8184	0.8667	5,921,000KB
RFE	0.8496	0.9336	0.9776	0.9920	0.9960	0.8429	0.9997	1,640,000KB
ADB	0.3552	0.4912	0.6360	0.7304	0.8352	0.1874	0.5829	3800KB
GBC	0.3760	0.5008	0.6496	0.7000	0.7536	0.3885	0.9992	6300KB
XGB	0.7888	0.8912	0.9416	0.9568	0.9728	0.7744	0.9987	6000KB

Table 19
ILSVRC-2012 final classification results on inception ResNet V2 intermediate data.

Model	Top-1 test accuracy	Top-2 test accuracy	Top-5 test accuracy	Top-10 test accuracy	Top-20 test accuracy	Average validation accuracy	Average training accuracy	Model size
MLP	0.8424	0.9248	0.9568	0.9704	0.9816	0.8459	0.9211	201,500KB
SVM	0.7360	0.8184	0.8512	0.8736	0.9136	0.7334	0.9997	984,000KB
LR	0.8712	0.9472	0.9768	0.9960	0.9992	0.8632	0.9957	70,100KB
KNN	0.8504	0.9344	0.9528	0.9568	0.9600	0.8451	0.8816	78,600KB
RFE	0.8536	0.9384	0.9792	0.9856	0.9896	0.8629	0.9997	1,530,000KB
ADB	0.4296	0.5512	0.6720	0.7544	0.8256	0.2347	0.5965	3800KB
GBC	0.5056	0.5976	0.6776	0.7192	0.7776	0.4549	0.9997	6200KB
XGB	0.8216	0.9056	0.9456	0.9608	0.9728	0.8117	0.9984	4200KB

scores. SVMs have the highest top-5 accuracy on every intermediate dataset. Interestingly, MLPs perform better than LR in top-5 accuracy on every dataset, although the same is not always the case for top-1 accuracy. For larger values of *n*, the top-*n* accuracy of KNN gets worse in comparison to other models.

4.3.3. ILSVRC-2012 results

The results of the classification algorithms on the intermediate datasets from Inception V3, Xception and Inception ResNet V2 on the ILSVRC-2012 subset can be found in Tables 17 respectively. The corresponding model configurations for the best performing models of each architecture can be found in Tables I.1, I.2 and I.3 in Appendix. The significance matrices on the validation data benchmarks are presented in Tables F.1, F.2 and F.3.

Because the original models for Inception V3, Xception and Inception ResNet were trained on 1000 classes, there is no benchmark for the CNN performance on the ILSVRC-2012 dataset with 100 classes as used in this paper. Therefore, the other classification models can only be compared to the respective MLP models for the intermediate datasets.

To summarise the results reported in Tables 17–19, on ILSVRC-2012 intermediate datasets, LR constantly reaches the highest accuracy results among the benchmarked models. LR outperforms MLPs by about 3 percentage points for top-1 accuracy or more on all ILSVRC-2012 intermediate datasets. The same observation is true for top-5 accuracy, although the minimal difference between MLPs and LR is 2 percentage points on this measure in Table 19. Moreover, the LR models are much smaller than the corresponding MLP models, thus enabling a reduction of size and better performance. Apart from LR, the RFE model also manages to outperform MLPs but with much larger model sizes. For example, the RFE model reaches an accuracy of 0.8536 on ResNet V2 intermediate data with a model size of 1,530,000KB. The corresponding MLP model is more than seven times smaller at 201,500KB. The performances of KNN models vary. On Inception ResNet V2 and Xception intermediate data, KNN is the third best model on top-1 test accuracy, outperforming MLPs. Interestingly, the performance on other metrics such as top-5 test accuracy is not as good for KNN as for other models. KNN models do not improve that much between top-1 and higher values of n. To illustrate this, on Inception V3 in Table 17, KNN performs almost 10 percentage points better than ADB regarding top-1 test accuracy with a score of 0.76608 for KNN and 0.6629 for ADB. On top-10 test accuracy, ADB performs about 10 percentage points better than KNN with a performance of 0.8938 versus 0.7904. The same pattern can be observed between other models and KNN and on other intermediate datasets. On every ILSVRC-2012 intermediate dataset, bagging manages to reach accuracy values of at least 3 percentage points higher than any of the boosting algorithms. Among the boosting approaches, XGB is consistently more proficient than ADB and GBCs, outperforming the latter by more than 30 percentage points. GBC still consistently performs better than ADB on all datasets. The ADB model's top-1 test accuracy was less than half of the score of the best performing model (LR) on every dataset. On all datasets, the accuracy on validation data is similar to the test data. This shows that the image filters from the benchmarked CNNs are not overfitted on the validation dataset.

4.4. Results interpretation and guidelines

This section provides an interpretation of the previously observed and described results, and it further analyses under which conditions specific classification algorithms outperform MLPs and can be considered superior to the traditional approach. Lastly, the section reveals which initial and intermediate datasets are more suitable for testing different classification algorithms. The main objective is to derive a guideline of when to test which classification algorithms on high-level image features, based on the observations from Section 4.3. The evaluation is based on the top-*n* test accuracy values and the model sizes.[19]

First of all, Section 4.2 proves that separately training the fully connected layers does not worsen the overall results, neither does it lead to an improvement in classification accuracy. It follows that the performance during the separated learning process is only dependent on the chosen classifier and is not worse than a unified learning procedure. Among the benchmark classifiers, the findings of Section 4.3 reveal a set of Pareto-optimal algorithms over all datasets. The Pareto-optimum is derived with respect to top-1 test accuracy and model size. Top-5 test accuracy is not used to build this set because some models randomly reach slightly higher top-5 accuracy values. For example, GBCs would only be Pareto-optimal on CNN-1 intermediate data from CIFAR-10 because of a marginally higher top-5 accuracy than LR. This would make the set of Pareto-optimal classifiers noisier.

On CIFAR-10 intermediate datasets, the set of Pareto-optimal classifiers is MLPs, LR, SVMs, and RFEs. On CIFAR-100 intermediate datasets, the set of classifiers is SVMs, LR, MLPs, and ADB and on the ILSVRC-2012 intermediate datasets, it is LR, ADB and XGB.

The following analysis of classification algorithms will focus on the union of these Pareto-optimal sets: LR, SVMs, MLPs, XGB, RFEs, and KNN. These algorithms have unique characteristics that can be reasons for applying them to high-level image features. ADB is discarded from the comparison. ADB is only Pareto-optimal on some intermediate datasets because of its small model sizes, but ADB consistently underperforms on top-*n* accuracy scores on all datasets. The following unique features can be identified among the models:

- **Logistic regression:** LR produces very small models that usually have very high and consistent top-*n* accuracy values
- **Support vector machines:** The overall top-*n* accuracy of SVMs varies. On certain intermediate datasets, SVMs are very good. SVMs are the best classifier on all intermediate datasets from CIFAR-100. On the other hand, SVMs seem to perform worse for datasets with more input features, such as the intermediate datasets from ILSVRC-2012. Additionally, the produced models are up to 100 times larger than the ones from MLPs.
- **Multi layer perceptron:** An MLP consistently has good performance values among the top performing models with moderate model sizes. On some benchmarks, MLPs even outperform all other models on top-1 accuracy (see Table 9). An LR model can, in theory, be reproduced by an MLP model. Nonetheless, LR models can be advantageous because they have fewer parameters to learn. Although this takes away complexity and limits the ability to detect more non-linear structures, this can be positive for fewer training samples.
- **eXtreme gradient boosting:** XGB produces reasonable top-*n* accuracies at small model sizes. In most cases, both top-*n* accuracy and model size are worse than LR, but XGB manages to perform better on some intermediate datasets. XGB can be proficient on many input features and output classes, because the produced model is still relatively small. For example, XGB keeps the model size very small on ILSVRC-2012 intermediate datasets while achieving reasonable accuracy scores.
- **Random forests estimator:** The only benchmark run where the RFE is Pareto-optimal is on the VGG-19 intermediate dataset from CIFAR-10. This superiority is caused by the small model size. Although a small model size is an atypical characteristic of an RFE model, the RFE was still included in this set of optimal classifiers because it can be superior to the benchmark model, MLP, in many cases (e.g., on all ILSVRC-2012 intermediate benchmarks).
- **K-nearest neighbours:** KNN produces very good top-1 accuracies on some benchmark datasets. Nonetheless, the top-*n*-accuracies for higher values of n are much worse than the compared classification algorithms. This indicates that the KNN model has more trouble detecting similarities among images because similar images should be more likely to be included in the top-*n* accuracy scores. KNN typically produces very large models.

Regarding model properties, the 11 benchmarked intermediate datasets differentiated between their number of input features and output classes. While CIFAR-10 produces 10 output classes, CIFAR-100 and the subsample of ILSVRC-2012 predict among 100 possible classes. For CIFAR-10 and CIFAR-100, the intermediate datasets had comparably few input features. SimpleNet intermediate data has the lowest number of input features (256 features), followed by VGG-19 intermediate data with 512 features. CNN-1 intermediate data has 2048 input features, and CNN-2 intermediate data has 2304. In comparison, the intermediate datasets from ILSVRC-2012 have much more input features. The Inception ResNet V2 intermediate dataset has 98,304 input features, the Inception V3 intermediate dataset has 131,072 input features, and the Xception intermediate dataset has 204,800 input features. This provides four intermediate datasets with relatively few input features and few output classes (CIFAR-10 intermediate datasets), four intermediate datasets with relatively few input features and many output classes (CIFAR-100 intermediate data), and three intermediate datasets with relatively many input features and many output classes (ILSVRC-2012 intermediate datasets). The following observations were made:

- **CIFAR-10 intermediate dataset:** On CIFAR-10 intermediate datasets, the initial CNN model's top-1 test accuracy was only outperformed on the SimpleNet intermediate dataset. Nonetheless, the differences between the MLP and the outperforming models are rather small, with the MLP only performing 0.2% worse than the best model (KNN). Apart from the SimpleNet intermediate dataset, the retrained MLP benchmark is only outperformed by SVMs on the CNN-2 intermediate dataset.
- **CIFAR-100 intermediate dataset:** On CIFAR-100 intermediate datasets, the CNN model's top-1 test accuracy is beaten on every intermediate dataset, apart from CNN-2. Many models outperform the CNN benchmark on SimpleNet and VGG-19 intermediate datasets, including MLPs, SVMs, LR, KNN, and RFEs. The MLP model is outperformed on every intermediate dataset, although the differences on SimpleNet and VGG-19 intermediate datasets are comparably smaller. In general, the accuracy values on those intermediate datasets are closer. SVMs have the highest top-1 test accuracy on every intermediate dataset.
- **ILSVRC-2012 intermediate dataset:** On ILSVRC-2012 intermediate datasets, LR and RFEs outperform the MLP benchmark on every dataset, while LR still performs better than RFEs. Additionally, KNN outperforms MLPs on top-1 test

[19] The model size depends on the choice of parameters and is therefore only an indicator of the general complexity of a given classification algorithm. As can be seen in the observations from the previous section, for some classification algorithms, the model size has a high variance depending on the chosen parameters and the dataset.

accuracy on the Xception and Inception ResNet V2 intermediate datasets. SVMs, on the other hand, perform much worse on all intermediate datasets compared to previous performance on CIFAR-10 and CIFAR-100 intermediate datasets.

From the above descriptions, one can derive a guideline to choose the best classification algorithms, depending on the intermediate dataset attributes. For intermediate datasets with few high-level image features as inputs and few output classes, such as CIFAR-10 intermediate datasets, an MLP seems to be proficient or equal to the other benchmarked algorithms. LR is a good choice if the model size needs to be kept small. On intermediate datasets with few output classes and relatively many input features, such as the CNN-2 and CNN-1 intermediate datasets, SVMs are able to perform better than MLPs.

For datasets with few input features and many output classes, such as CIFAR-100 intermediate datasets, it is advisable to use classification algorithms other than an MLP. The best choice of model is SVMs, although this comes at the cost of large model sizes. For the models with fewer input features (SimpleNet, VGG-19), the choice of the final classifier does not have such a big impact on top-*n* accuracy. On the other hand, the effect of the model choice on the final performance is larger for intermediate datasets with more input features (CNN-1, CNN-2). The superior performance of the benchmarked classification algorithms over an MLP could also be caused by the fewer training samples (600 samples per class compared to 6000 samples per class for CIFAR-10). For example, SVMs generally learn well on small datasets.

For datasets with many input features and many output classes, such as ILSVRC-2012 intermediate datasets, more proficient models than MLP are available. The LR and RFE models consistently outperform MLPs in the benchmark and can thus be considered superior for these datasets. LR has the additional advantage of small model sizes whereas RFEs should not be used if model complexity is an issue. However, even some of the other benchmarked models, including KNN and XGB, should be considered to replace MLPs because they can be advantageous depending on the problem at hand.

Overall, other classification algorithms are usually more proficient than MLPs if the intermediate dataset has many input features or if the classification problem has many output classes. Another decisive factor can be the number of samples per class for the classifier to learn from. The number of output classes is determined by the problem at hand. The number of input features from the intermediate dataset can be controlled for in the configuration of the CNN that is used to create the intermediate dataset. Therefore, whether fewer high-level output features are created is a design choice, and a higher likelihood of an MLP being the best choice or more output features in combination with other classification algorithms is desirable.

Apart from the established guidelines, it is always good practice to benchmark as many different classification algorithms and setups as possible, subject to time and computational constraints.

5. Conclusion

Over the course of research on CNN architectures, few modifications have been made to the fully connected layers at the end of the networks. In image classification, these neural network layers are responsible for creating the final classification results based on the output of the last layer of high-level image filters. Traditionally, this final classification is performed with fully connected neural network layers (MLPs). The advantage is that these models can be integrated into the learning process of CNNs because both can use backpropagation as their training procedure. The choice of the best classifier for a given problem is not a deterministic decision. While the overall CNN architecture, independently from the final classifier, plays a fundamental role in the final classification performance and is responsible for extracting high-level image features, one can choose different setups for the final classification step of the high-level image features.

To investigate this aspect, this paper proposes a two-step approach to assess the possibility of improving the performance of computer vision classification models by using different classification algorithms on high-level image features. In the first step, different CNN configurations are trained on several benchmark datasets. In the second step, the fully connected neural network layers that make the final prediction are removed from the CNNs, such that the last remaining layer produces a flattened output from the last layer of image filters. On this intermediate output, multiple classification algorithms are benchmarked.

Experimental results from different datasets and CNN architectures showed that other classification algorithms, namely LR, SVMs, XGB, RFEs, and KNN, have unique characteristics that can lead to a better performance than MLP models. In particular, the likelihood of a model outperforming an MLP model is higher for high-dimensional intermediate datasets. However, setups with many output classes, as is the case in the CIFAR-100 intermediate models, also proved to have better classification accuracy with models other than MLPs. While the classification algorithms largely perform on par with MLPs and the initial CNN benchmark on CIFAR-10, some models clearly perform better than those two benchmarks on CIFAR-100 or ILSVRC-2012. Particularly, LR and SVMs show strong performances throughout the benchmark. While LR works well for high-dimensional datasets (e.g., the intermediate datasets from ILSVRC-2012), SVMs work better for lower-dimensional datasets with many output classes (e.g., CIFAR-100). LR produces among the smallest models in the benchmarks, while the SVM models are rather large. From the remaining classifiers with unique characteristics, RFEs and KNN tend to produce very large models. KNN also gets worse on top-*n* accuracy for higher values of n. The used configurations of XGB have the advantage of producing small model sizes, even for high-dimensional datasets (e.g., on the ILSVRC-2012 intermediate datasets). For datasets with very few input features (e.g., SimpleNet and VGG-19 intermediate data from CIFAR-10 and CIFAR-100), the final classification performance is less reliant on the actual classification algorithm used in the end. All in all, it is recommended to benchmark other classification algorithms on the produced high-level image features.

The insights from this paper offer various opportunities for further research. This paper focuses on the classification task in computer vision and shows that replacing the fully-connected neural network layers with a different classifier could enhance the classification performance. This insight alone is valuable for further research and for industry applications. Nonetheless, the scope of these findings can be expanded from image classification to the classification step of the other computer vision tasks introduced in Section 1. Furthermore, the same procedure can be applied to regression tasks instead of classification, thus replacing the final fully-connected layers with classification algorithms that produce continuous outputs.

Additionally, the set of benchmark classification algorithms can be enlarged to include more classification algorithms to evaluate their predictive capability on high-level image features, i.e., the benchmark in this paper did not look at evolutionary algorithms.

Another future contribution could be the unification of the two-step procedure described in this paper. The classification algorithm from the second step would have to be integrated into the overall learning process of the CNN. This is easily implementable for classification algorithms that make use of an iterative process to optimize the value of a loss function. Tang (2013) exemplifies how this can be done for linear SVMs optimising a margin-based loss function.

Moreover, from an input features perspective, it would be an interesting approach to use image filters from multiple layers. In this case, the classification algorithm would not only learn on the last layer of image features but can also make predictions on lower-level image features, if they contain any unique information.

Regarding the structure of the convolutional layers to extract the image features, the findings from the work of Jagusch, Gonçalves, and Castelli (2018) can be useful to determine effective network structures to build up feature extractors. The authors use an evolutionary approach to build up neural network structures over multiple generations (neuroevolution). The most interesting characteristic of this approach, called semantic learning machine (SLM), is that it searches over unimodal error landscapes in any supervised learning problem where the error is measured as a distance to the known targets. This means, with the exception of the global optimum, every point in the search space has at least one neighbor with better fitness, and that neighbor is reachable through the application of the variation operators. The first application of the SLM in the field of image analysis appeared in Lapa, Gonçalves, Rundo, and Castelli (2019), where authors showed that the SLM outperforms a state-of-the-art CNN trained with back-propagation on the classification of high-resolution multiparametric Magnetic Resonance Imaging with statistical significance. While the work presented in Lapa et al. (2019) tries to optimize the topology of the fully connected network part, this idea can be extended to the convolutional layers and, subsequently, to the optimization of the hyperparameters of the CNN.

The literature review in Section 2.1 reveals that several authors chose other classifiers than fully-connected neural networks for the final classification because of the limited number of training data. While the scarceness of training data was not an issue for the datasets considered in this study, it could be interesting to investigate the performance of the different algorithms when varying the size of the input datasets.

Lastly, in current research, the extraction of universal image features is commonly used in transfer learning environments. In these applications, the convolutional layers from one network trained on a given dataset are frozen and applied to a different dataset where the network is trained further or only the last classifier is retrained. It would be interesting to investigate whether different classifiers can outperform neural networks in this environment.

Declaration of Competing Interest

Authors declare no conflict of interests.

Acknowledgments

This work was supported by national funds through Fundação para a Ciência e a Tecnologia under project DSAIPA/DS/0022/2018 (GADgET). Authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P5-0410).

Appendix A. Visualisation of the steps of the proposed approach

Figs. A.1, A.2, and A.3 show the steps of the proposed approach.

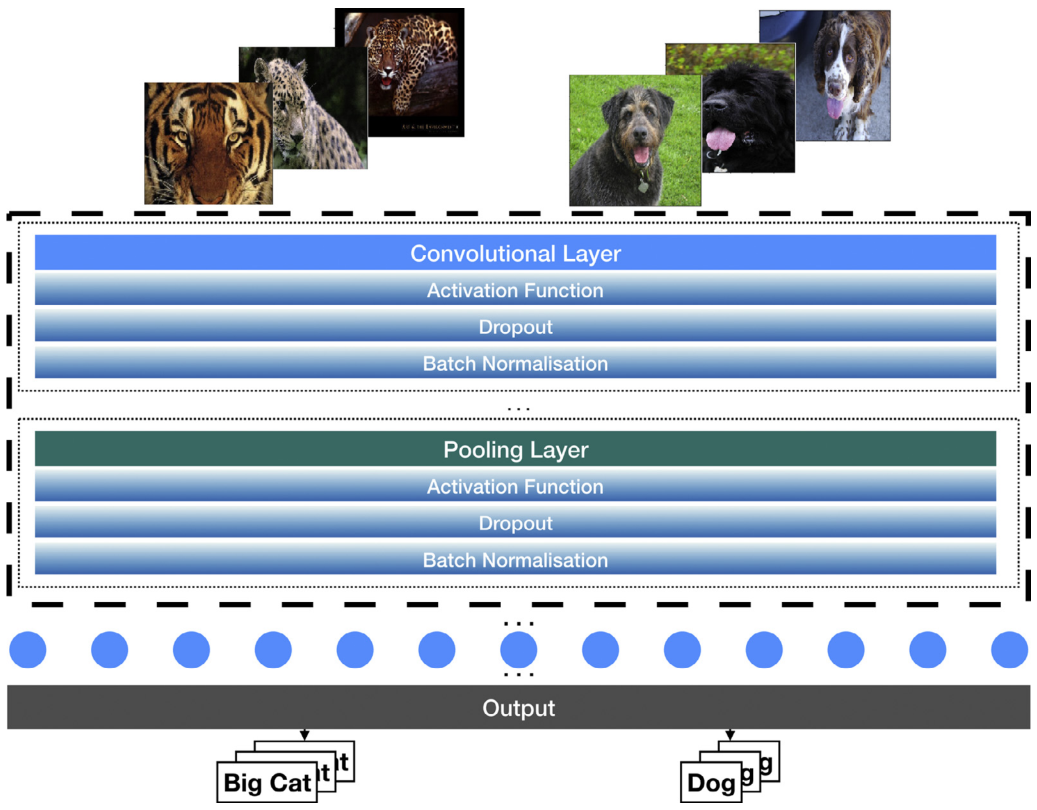


Fig. A.1. Visualisation of the general architecture of a CNN.

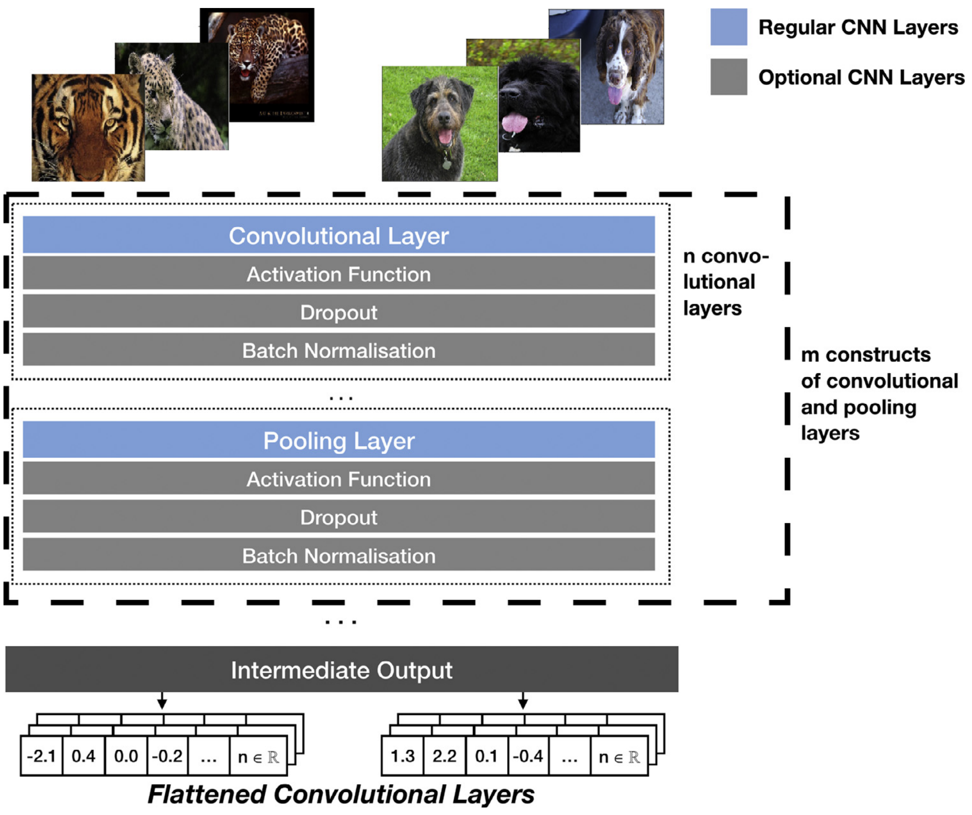
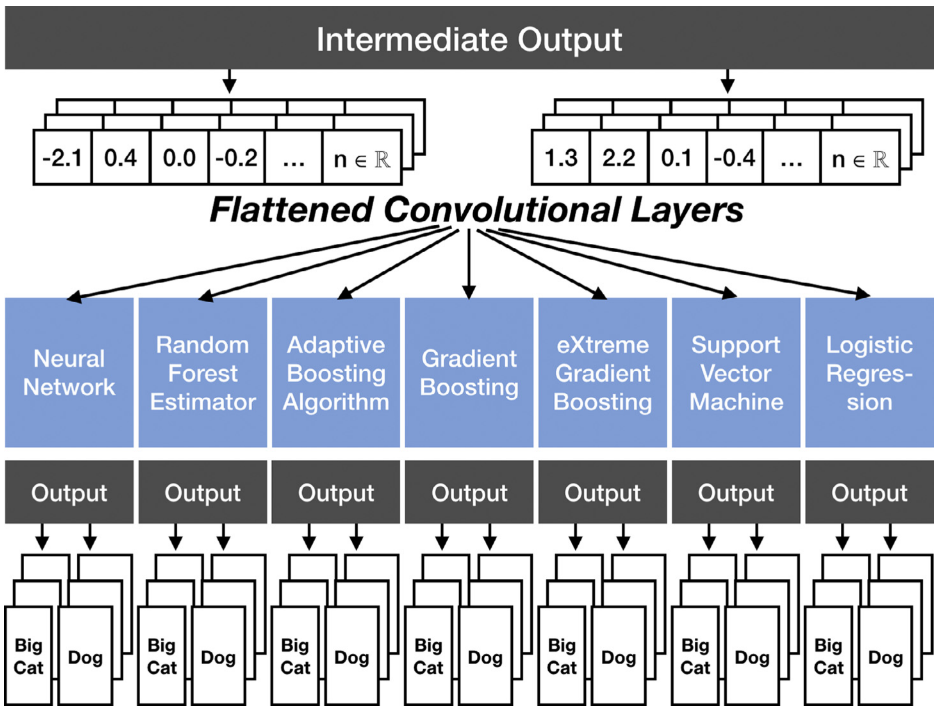


Fig. A.2. Visualisation of intermediate data creation.



Intermediate Output
-2.1 0.4 0.0 -0.2 ... n ∈ ℝ
1.3 2.2 0.1 -0.4 ... n ∈ ℝ
Flattened Convolutional Layers
Neural Network
Random Forest Estimator
Adaptive Boosting Algorithm
Gradient Boosting
eXtreme Gradient Boosting
Support Vector Machine
Logistic Regres-sion
Output
Big Cat
Dog

Fig. A.3. Visualisation of intermediate data classification.

Appendix B. External dataset benchmarks

Table B.1
CIFAR-10 benchmarks.

Model	Source	Param	Accuracy
SimpleNet	HasanPour et al. (2016)	5.48M	95.32 %
SD-110L	Huang, Sun, Liu, Sedra, and Weinberger (2016)	1.7M	94.77 %
VGG-19 (local benchmark)	Simonyan and Zisserman (2014)	15M	93.59 %
WRN	Zagoruyko and Komodakis (2016)	600K	93.15 %
ALLCNN	Springenberg, Dosovitskiy, Brox, and Riedmiller (2014)	1.3M	92.75 %
DSN	Lee, Xie, Gallagher, Zhang, and Tu (2015)	1M	92.03 %
FitNet	Srivastava, Greff, and Schmidhuber (2015)	1M	91.61 %
ResNet-32 (depth of 32)	HasanPour et al. (2016)	475K	91.60 %
NiN	Lin, Chen, and Yan (2013)	1M	91.19 %
dasNet	Stollenga, Masci, Gomez, and Schmidhuber (2014)	6M	90.78 %
Maxout (k = 2)	Goodfellow, Warde-Farley, Mirza, Courville, and Bengio (2013)	6M	90.62 %
SimpleNet (local benchmark)	HasanPour et al. (2016)	5.48M	88.52 %

Table B.2
CIFAR-100 benchmarks.

Model	Source	Param	Accuracy
SD-110L	Huang et al. (2016)	1.7m	75.42 %
SimpleNet	HasanPour et al. (2016)	5.48M	73.42 %
VGG-19 (local benchmark)	Simonyan and Zisserman (2014)	15M	70.48 %
WRN	Zagoruyko and Komodakis (2016)	600K	69.11 %
ResNet-32 (depth of 32)	HasanPour et al. (2016)	475K	67.37 %
ALLCNN	Springenberg et al. (2014)	1.3M	66.29 %
dasNet	Stollenga et al. (2014)	6M	66.22 %
Maxout (k = 2)	Goodfellow et al. (2013)	6M	65.46 %
DSN	Lee et al. (2015)	1M	65.43 %
FitNet	Srivastava et al. (2015)	1M	64.96 %
NiN	Lin et al. (2013)	1M	64.32 %
SimpleNet (local benchmark)	HasanPour et al. (2016)	5.48M	60.99 %

Table B.3
ILSVRC 2012 benchmarks.

Model	Source	Top-5 accuracy
Inception ResNet V2	Szegedy et al. (2017)	95.1 %
Xception	Chollet (2016)	94.5 %
Inception V3	Szegedy et al. (2016)	94.4 %
ResNet-152	He et al. (2016)	92.9 %
ResNet-101	He et al. (2016)	92.6 %
ResNet-50	He et al. (2016)	92.0 %
VGG-16	Simonyan and Zisserman (2014)	89.9 %
GoogLeNet	Szegedy et al. (2015)	89.1 %
Network in Network	Lin et al. (2013)	81.2 %
CaffeNet	Jia et al. (2014)	79.9 %
AlexNet	Krizhevsky et al. (2012)	79.8 %

Appendix C. Learning curves for CNN networks

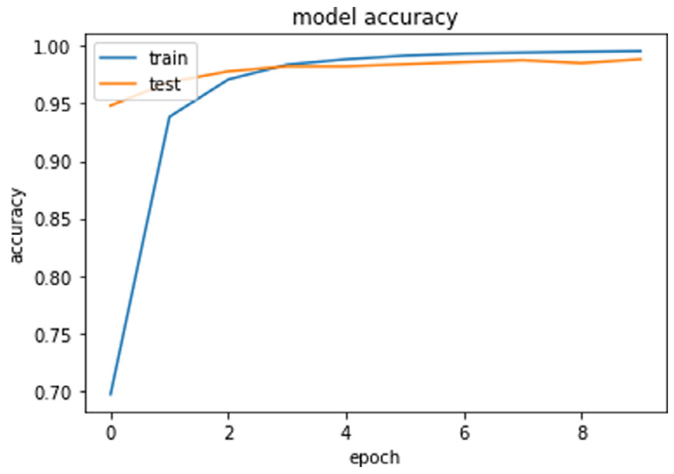


Fig. C.1. Learning curve of MLP-0 on CNN-1 intermediate data for 10 iterations with learning rate 0.0001.

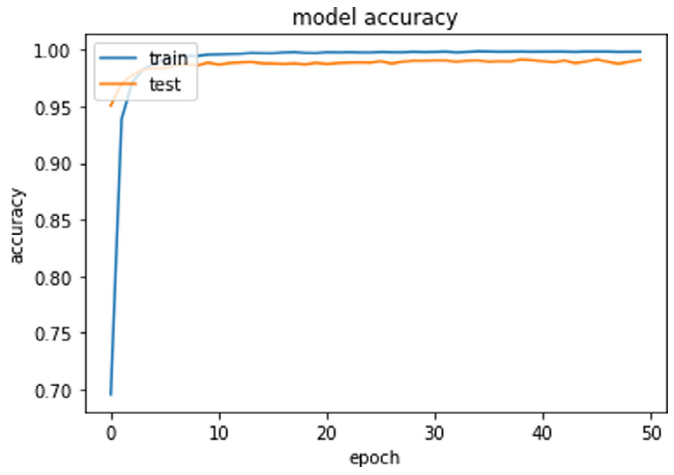


Fig. C.2. Learning curve of MLP-0 on CNN-1 intermediate data for 50 iterations with learning rate 0.0001.

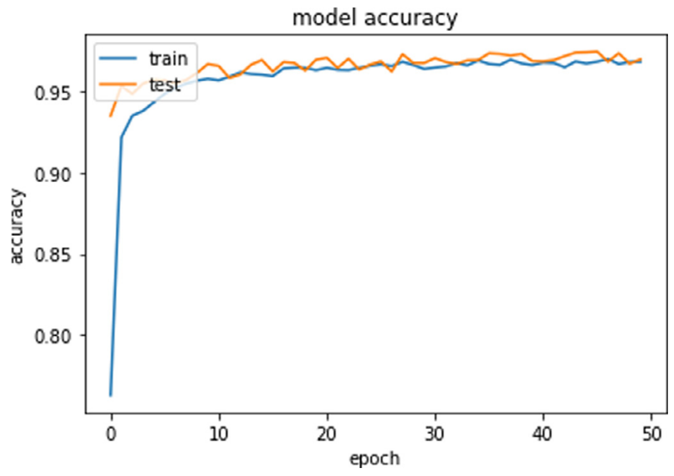


Fig. C.3. Learning curve of MLP-0 on CNN-1 intermediate data for 50 iterations with learning rate 0.001.

Appendix D. CIFAR-10 significance tests

In the diagonal lines of the tables, the mean accuracy of the validation data over 10 cross-validation folds is displayed. In the other cells, the *p*-value of the comparison of the two indicated

Table D.1
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for MLP-1, MLP-2, MLP-3 on CNN-1 intermediate data from CIFAR-10 with different learning rates (LR). Values lower than 10^{-4} reported as 0.

	MLP-1_LR-0.0001	MLP-2_LR-0.0001	MLP-3_LR-0.0001	MLP-1_LR-0.001	MLP-2_LR-0.001	MLP-3_LR-0.001
MLP-1_LR-0.0001	**0.9987**	1.00				
MLP-2_LR-0.0001	1.00	**0.9987**				
MLP-3_LR-0.0001	0.0004	0.0004	**0.9913**	0.0078	0.0017	0.0007
MLP-1_LR-0.001	0	0		**0.9948**	0.0043	0
MLP-2_LR-0.001	0.1651	0.1608			**0.9974**	0.5691
MLP-3_LR-0.001	0.0037	0.0002				**0.9979**

Table D.2
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for MLP-1, MLP-2, MLP-3 on CNN-2 intermediate data from CIFAR-10 with different learning rates (LR). Values lower than 10^{-4} reported as 0.

	MLP-1_LR-0.0001	MLP-2_LR-0.0001	MLP-3_LR-0.0001	MLP-1_LR-0.001	MLP-2_LR-0.001	MLP-3_LR-0.001
MLP-1_LR-0.0001	**0.7781**	0.0491			0.8565	
MLP-2_LR-0.0001		**0.7908**				
MLP-3_LR-0.0001	0.0172	0.0004	**0.7638**	0.0021	0.0011	
MLP-1_LR-0.001	0.3577	0.0027		**0.7742**	0.0315	
MLP-2_LR-0.001		0.0195			**0.7789**	
MLP-3_LR-0.001	0	0	0.0001	0	0	**0.7364**

Table D.3
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for MLP-1, MLP-2, MLP-3 on SimpleNet intermediate data from CIFAR-10 with different learning rates (LR). Values lower than 10^{-4} reported as 0.

	MLP-1_LR-0.0001	MLP-2_LR-0.0001	MLP-3_LR-0.0001	MLP-1_LR-0.001	MLP-2_LR-0.001	MLP-3_LR-0.001
MLP-1_LR-0.0001	**0.9993**					1
MLP-2_LR-0.0001	0.4923	**0.9992**				0.6016
MLP-3_LR-0.0001	0.0315	0.1494	**0.9990**			0.1143
MLP-1_LR-0.001	0.0888	0.117	0.1933	**0.9983**	0.3623	0.0964
MLP-2_LR-0.001	0.2169	0.3136	0.591		**0.9988**	0.2381
MLP-3_LR-0.001	1					**0.9993**

Table D.4
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for MLP-1, MLP-2, MLP-3 on VGG-19 intermediate data from CIFAR-10 with different learning rates (LR). Values lower than 10^{-4} reported as 0.

	MLP-1_LR-0.0001	MLP-2_LR-0.0001	MLP-3_LR-0.0001	MLP-1_LR-0.001	MLP-2_LR-0.001	MLP-3_LR-0.001
MLP-1_LR-0.0001	**0.9999**	0.8135	1	1		1
MLP-2_LR-0.0001		**0.9999**				
MLP-3_LR-0.0001	1	0.8135	**0.9999**	1		1
MLP-1_LR-0.001	1	0.6328	1	**0.9999**		1
MLP-2_LR-0.001	0.6382	0.5465	0.6382	0.6272	**0.9998**	0.6272
MLP-3_LR-0.001	1	0.6328	1	1		**0.9999**

Table D.5
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for different classification algorithms on CNN-1 intermediate data from CIFAR-10. Values lower than 10^{-4} reported as 0.

	MLP	SVM	LR	KNN	RFE	ADB	GBC	XGB
MLP	**0.9987**		0.0004					
SVM	0	**0.9819**	0					
LR			**0.9998**					
KNN	0	0	0	**0.8245**	0.0001	0.0001	0	0.0001
RFE	0	0	0		**0.9254**		0	0.0001
ADB	0	0	0		0.0018	**0.9119**	0	0.0001
GBC	0.0001	0.0018	0.0001				**0.9722**	
XGB	0	0.0001	0				0.0004	**0.9527**

Table D.6
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for different classification algorithms on CNN-2 intermediate data from CIFAR-10. Values lower than 10^{-4} reported as 0.

	MLP	SVM	LR	KNN	RFE	ADB	GBC	XGB
MLP	**0.7908**							
SVM	0.2201	**0.7851**						
LR	0.0796	0.1822	**0.775**					
KNN	0	0	0	**0.5789**	0	0.0001	0	0
RFE	0	0	0.0001		**0.6962**		0.1331	0.0001
ADB	0	0	0		0.0001	**0.6456**	0.0001	0.
GBC	0	0	0.0001				**0.7015**	0.000379
XGB	0.0001	0	0.0004					**0.7331**

Table D.7
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for different classification algorithms on SimpleNet intermediate data from CIFAR-10. Values lower than 10^{-4} reported as 0.

	MLP	SVM	LR	KNN	RFE	ADB	GBC	XGB
MLP	**0.9994**	0.309181	0.000622					
SVM		**0.9995**	0.000276					
LR			**1.0000**					
KNN	0.0731	0.0601	0.0188	**0.9982**	0.5655			
RFE	0.0007	0.0004	0.0001		**0.9986**			
ADB	0.0009	0.0009	0.0007	0.0017	0.0013	**0.9909**	0.002936	
GBC	0.0005	0.0004	0.0001	0.7883	0.0357		**0.9982**	
XGB	0	0	0	0.011	0.0001		0.0003	**0.9971**

Table D.8
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for different classification algorithms on VGG-19 intermediate data from CIFAR-10. Values lower than 10^{-4} reported as 0.

	MLP	SVM	LR	KNN	RFE	ADB	GBC	XGB
MLP	**0.9999**	1	1	1	1			
SVM	1	**0.9999**	1	1	1			
LR	1	1	**0.9999**	1	1			
KNN	1	1	1	**0.9999**	1			
RFE	1	1	1	1	**0.9999**			
ADB	0.3606	0.3701	0.3651	0.3828	0.382835	**0.9998**		1
GBC	0.0313	0.0337	0.0324	0.0376	0.0372	0.1305	**0.9996**	0.1108
XGB	0.2263	0.2424	0.234	0.2663	0.2637	1		**0.9998**

networks/techniques is given. Since the comparisons are symmetric, the *p*-value is only displayed for those values where the estimated population mean of the network/technique in the column name is higher than the estimated population mean of the network/technique in the row name that it is compared to. The other corresponding cell is left empty.

Appendix E. CIFAR-100 significance tests

In the diagonal lines of the tables, the mean accuracy of the validation data over 10 cross-validation folds is displayed. In the other cells, the *p*-value of the comparison of the two indicated networks/techniques is given. Since the comparisons are symmetric, the *p*-value is only displayed for those values where the estimated population mean of the network/technique in the column name is higher than the estimated population mean of the network/technique in the row name that it is compared to. The other corresponding cell is left empty.

Table E.1
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for MLP-1, MLP-2, MLP-3 on CNN-1 intermediate data from CIFAR-100 with different learning rates (LR). Values lower than 10^{-4} reported as 0.

	MLP-1_LR-0.0001	MLP-2_LR-0.0001	MLP-3_LR-0.0001	MLP-1_LR-0.001	MLP-2_LR-0.001	MLP-3_LR-0.001
MLP-1_LR-0.0001	**0.9896**					
MLP-2_LR-0.0001	0.4952	**0.9884**	0.911870			
MLP-3_LR-0.0001	0.768		**0.9887**			
MLP-1_LR-0.001	0.0001	0.0001	0.0007	**0.9680**	0.1172	0.9643
MLP-2_LR-0.001	0.0009	0.0027	0.0069		**0.9738**	
MLP-3_LR-0.001	0.0001	0.0001	0.0007		0.111	**0.9681**

Table E.2
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for MLP-1, MLP-2, MLP-3 on CNN-2 intermediate data from CIFAR-100 with different learning rates (LR). Values lower than 10^{-4} reported as 0.

	MLP-1_LR-0.0001	MLP-2_LR-0.0001	MLP-3_LR-0.0001	MLP-1_LR-0.001	MLP-2_LR-0.001	MLP-3_LR-0.001
MLP-1_LR-0.0001	**0.4494**					
MLP-2_LR-0.0001	0.048	**0.4402**				
MLP-3_LR-0.0001	0.0004	0.0092	**0.4235**			
MLP-1_LR-0.001	0.0001	0.0004	0.0191	**0.4102**		
MLP-2_LR-0.001	0	0.0001	0.0001	0.0004	**0.3762**	
MLP-3_LR-0.001	0	0	0	0.0001	0.2668	**0.3713**

Table E.3
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for MLP-1, MLP-2, MLP-3 on SimpleNet intermediate data from CIFAR-100 with different learning rates (LR). Values lower than 10^{-4} reported as 0.

	MLP-1_LR-0.0001	MLP-2_LR-0.0001	MLP-3_LR-0.0001	MLP-1_LR-0.001	MLP-2_LR-0.001	MLP-3_LR-0.001
MLP-1_LR-0.0001	**0.9838**					
MLP-2_LR-0.0001	0.0001	**0.9589**				
MLP-3_LR-0.0001	0	0.0005	**0.9417**	0.7857		0.0642
MLP-1_LR-0.001	0.0001	0.0103		**0.943**		0.1233
MLP-2_LR-0.001	0	0.0002	0.01404	0.0296	**0.9275**	0.0067
MLP-3_LR-0.001	0.0034	0.6226				**0.9557**

Table E.4
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for MLP-1, MLP-2, MLP-3 on VGG-19 from CIFAR-100 intermediate data with different learning rates (LR),. Values lower than 10^{-4} reported as 0.

	MLP-1_LR-0.0001	MLP-2_LR-0.0001	MLP-3_LR-0.0001	MLP-1_LR-0.001	MLP-2_LR-0.001	MLP-3_LR-0.001
MLP-1_LR-0.0001	**0.9962**	0.1436	0.0464			
MLP-2_LR-0.0001		**0.9968**	0.63			
MLP-3_LR-0.0001			**0.9969**			
MLP-1_LR-0.001	0.0006	0.0001	0	**0.9940**	0.0078	0.1228
MLP-2_LR-0.001	0.134	0.0046	0.0007		**0.9955**	
MLP-3_LR-0.001	0.0008	0	0		0.037	**0.9948**

Table E.5
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for different classification algorithms on CNN-1 intermediate data from CIFAR-100. Values lower than 10^{-4} reported as 0.

	MLP	SVM	LR	KNN	RFE	ADB	GBC	XGB
MLP	**0.9884**		0.017690					
SVM	0.011099	**0.9820**	0.000015					
LR			**0.9935**					
KNN	0	0	0	**0.3769**			0.0009	0
RFE	0	0	0	0.0028	**0.3576**		0.0002	0
ADB	0	0	0	0	0	**0.2482**	0	0
GBC	0	0	0				**0.4369**	0.0097
XGB	0	0	0					**0.4721**

Table E.6
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for different classification algorithms on CNN-2 intermediate data from CIFAR-100.Values lower than 10^{-4} reported as 0.

	MLP	SVM	LR	KNN	RFE	ADB	GBC	XGB
MLP	**0.4494**	0.0071						0
SVM		**0.4576**						0
LR	0.0113	0.0001	**0.4417**					0
KNN	0	0	0	**0.2666**	0.0172		0.0001	0
RFE	0	0	0		**0.2793**		0.0001	0
ADB	0	0	0	0	0	**0.1732**	0	0
GBC	0.0007	0.0003	0.0013				**0.3885**	0
XGB								**0.7744**

Table E.7
Pairwise *p*-values of performance on validation set with performance on validation set displayed on the diagonal for different classification algorithms on SimpleNet intermediate data from CIFAR-100. Values lower than 10^{-4} reported as 0.

	MLP	SVM	LR	KNN	RFE	ADB	GBC	XGB
MLP	**0.9838**		0.0001					
SVM	0.0317	**0.9801**	0					
LR			**0.9937**					
KNN	0	0	0	**0.8380**				
RFE	0	0	0	0.0002	**0.8258**			
ADB	0	0	0	0	0	**0.5165**		0
GBC	0	0	0	0	0	0.0001	**0.3885**	0
XGB	0	0	0	0	0			**0.7744**

Table E.8
Pairwise p-values of performance on validation set with performance on validation set displayed on the diagonal for different classification algorithms on VGG-19 intermediate data from CIFAR-100. Values lower than 10^{-4} reported as 0.

	MLP	SVM	LR	KNN	RFE	ADB	GBC	XGB
MLP	**0.9968**	0.5389	0.8449	0.7807	0.8372			
SVM		**0.9972**						
LR		0.5912	**0.9969**	0.9167	1			
KNN		0.6729		**0.997**				
RFE		0.4357	1	0.8895	**0.9969**			
ADB	0.0034	0.0028	0.0032	0.0031	0.0031	**0.9854**	0.0339	0.0065
GBC	0.0001	0	0.0001	0.0001	0.0001		**0.9911**	0.0001
XGB	0.0099	0.0009	0.0043	0.0039	0.00214			**0.9948**

Appendix F. ILSVRC-2012 significance tests

In the diagonal lines of the tables, the mean accuracy of the validation data over 10 cross-validation folds is displayed. In the other cells, the p-value of the comparison of the two indicated networks/techniques is given. Since the comparisons are symmetric, the p-value is only displayed for those values where the estimated population mean of the network/technique in the column name is higher than the estimated population mean of the network/technique in the row name that it is compared to. The other corresponding cell is left empty.

Table F.1
Pairwise p-values of performance on validation set with performance on validation set displayed on the diagonal for different classification algorithms on Inception ResNet V2 intermediate data from ILSVRC-2012 with 100 classes. Values lower than 10^{-4} reported as 0.

	MLP	SVM	LR	KNN	RFE	ADB	GBC	XGB
MLP	**0.8459**		0.0669		0.0386			
SVM	0.0048	**0.7334**	0.0048	0.0048	0.0042			0.0118
LR			**0.8632**					
KNN	0.6409		0.0613	**0.8451**	0.035			
RFE			0.9662		**0.8629**			
ADB	0.0003	0.0006	0.0003	0.0003	0.0003	**0.2347**	0.0085	0.0003
GBC	0.0021	0.0048	0.002	0.0021	0.0019		**0.4549**	0.0026
XGB	0.0131		0.0132	0.0136	0.0095			**0.8117**

Table F.2
Pairwise p-values of performance on validation set with performance on validation set displayed on the diagonal for different classification algorithms on Inception V3 intermediate data from ILSVRC-2012 with 100 classes. Values lower than 10^{-4} reported as 0.

	MLP	SVM	LR	KNN	RFE	ADB	GBC	XGB
MLP	**0.7760**		0.0136		0.014			
SVM	0.0007	**0.3168**	0.0003	0.0004	0.0003		0.0057	0.0006
LR			**0.8459**					
KNN	0.0903		0.0006	**0.7509**	0.0002			
RFE			0.0044		**0.8157**			
ADB	0.0007	0.0094	0.0005	0.0006	0.0005	**0.1341**	0.0026	0.0007
GBC	0.0008		0.0002	0.0003	0.0002		**0.4515**	0.0007
XGB	0.0923		0.0035	0.4908	0.0034			**0.7461**

Table F.3
Pairwise p-values of performance on validation set with performance on validation set displayed on the diagonal for different classification algorithms on Xception intermediate data from ILSVRC-2012 with 100 classes. Values lower than 10^{-4} reported as 0.

	MLP	SVM	LR	KNN	RFE	ADB	GBC	XGB
MLP	**0.8245**		0.0169		0.0295			
SVM	0.0146	**0.7483**	0.0073	0.0237	0.0085			0.0999
LR			**0.8528**					
KNN	0.4929		0.037	**0.8184**	0.0663			
RFE			0.037		**0.8429**			
ADB	0.0002	0.0005	0.0001	0.0003	0.0002	**0.1847**	0.0059	0.0002
GBC	0.001	0.0019	0.0008	0.0012	0.0008		**0.3885**	0.0012
XGB	0.0054		0.0011	0.0237	0.0008			**0.7744**

Appendix G. Best parameters for classification algorithms on CIFAR-10 intermediate data

Table G.1
Best parameters for classification algorithms on CIFAR-10 intermediate data on CNN-1.

model	best model parameters on test
MLP	MLP-2, LR: 0.0001
SVM	C: 10, γ: 0.001, kernel: rbf
LR	C: 0.001, penalty: $\ell 2$
KNN	neighbours: 10
RFE	1000 estimators, gini criterion
ADB	1000 estimators, LR: 0.5
GBC	100 estimators, LR: 0.5
XGB	100 estimators, LR: 0.1, max. depth: 5

Table G.2
Best parameters for classification algorithms on CIFAR-10 intermediate data on CNN-2.

Model	Best parameters
MLP	MLP-2, LR: 0.0001
SVM	C: 10, γ: 0.001, kernel: rbf
LR	C: 0.001, penalty: $\ell 2$
KNN	neighbours: 1
RFE	1000 estimators, gini criterion
ADB	1000 estimators, LR: 0.1
GBC	100 estimators, LR: 0.1
XGB	100 estimators, LR: 0.1, max. depth: 10

Table G.3
Best parameters for classification algorithms on CIFAR-10 intermediate data on SimpleNet.

Model	Best parameters
MLP	MLP-1, LR: 0.0001
SVM	C: 10, γ: 0.001, kernel: rbf
LR	C: 0.1, penalty: $\ell 2$
KNN	neighbours: 10
RFE	1000 estimators, entropy criterion
ADB	1000 estimators, LR: 0.1
GBC	100 estimators, LR: 0.1
XGB	100 estimators, LR: 0.1,max. depth: 5

Table G.4
Best parameters for classification algorithms on CIFAR-10 intermediate data on VGG-19.

Model	Best parameters
MLP	MLP-2, LR: 0.0001
SVM	C: 1, γ: 0.0001, kernel: rbf
LR	C: 0.001, penalty: $\ell 2$
KNN	neighbours: 10
RFE	100 estimators, gini criterion
ADB	1000 estimators, LR: 0.5
GBC	100 estimators, LR: 0.05
XGB	100 estimators, LR: 0.1, max. depth: 1

Appendix H. Best parameters for classification algorithms on CIFAR-100 intermediate data

Table H.1
Best parameters for classification algorithms on CIFAR-100 intermediate data on CNN-1.

Model	Best parameters
MLP	MLP-2, LR: 0.0001
SVM	C: 10, γ: 0.0001, kernel: rbf
LR	C: 0.1, penalty: $\ell 2$
KNN	neighbours: 1
RFE	100 estimators, entropy criterion
ADB	100 estimators, LR: 0.1
GBC	100 estimators, LR: 0.05
XGB	100 estimators, LR: 0.1, max. depth: 5

Table H.2
Best parameters for classification algorithms on CIFAR-100 intermediate data on CNN-2.

Model	Best parameters
MLP	MLP-1, LR: 0.0001
SVM	C: 10, γ: 0.0001, kernel: rbf
LR	C: 0.001, penalty: $\ell 2$
KNN	neighbours: 1
RFE	100 estimators, entropy criterion
ADB	100 estimators, LR: 0.1
GBC	100 estimators, LR: 0.1
XGB	100 estimators, LR: 0.1, max. depth: 1

Table H.3
Best parameters for classification algorithms on CIFAR-100 intermediate data on SimpleNet.

Model	Best parameters
MLP	MLP-1, LR: 0.0001
SVM	C: 10, γ: 0.0001, kernel: rbf
LR	C: 0.1, penalty: $\ell 2$
KNN	neighbours: 10
RFE	100 estimators, entropy criterion
ADB	100 estimators, LR: 0.1
GBC	100 estimators, LR: 0.05
XGB	100 estimators, LR: 0.1, max. depth: 5

Table H.4
Best parameters for classification algorithms on CIFAR-100 intermediate data on VGG-19.

Model	Best parameters
MLP	MLP-2, LR: 0.0001
SVM	C: 1, γ: 0.001, kernel: rbf
LR	C: 0.1, penalty: $\ell 2$
KNN	neighbours: 10
RFE	1000 estimators, gini criterion
ADB	100 estimators, LR: 0.1
GBC	100 estimators, LR: 0.05
XGB	100 estimators, LR: 0.1, max. depth: 5

Appendix I. Best parameters of classifiers on high-level image features

Table I.1
ILSVRC-2012: best parameters on Inception ResNet V2.

Model	Best parameters
MLP	MLP-2, LR: 0.0001
SVM	C: 10, γ: 0.0001, kernel: rbf
LR	C: 0.001, penalty: $\ell 2$
KNN	neighbours: 10
RFE	1000 estimators, gini criterion
ADB	1000 estimators, LR: 0.1
GBC	100 estimators, LR: 0.1
XGB	100 estimators, LR: 0.1, max. depth: 1

Table I.2
ILSVRC-2012 best parameters on Inception V3.

Model	Best parameters
MLP	MLP-1, LR: 0.0001
SVM	C: 10, γ: 0.0001, kernel: rbf
LR	C: 0.001, penalty: $\ell 2$
KNN	neighbours: 1
RFE	1000 estimators, entropy criterion
ADB	1000 estimators, LR: 0.1
GBC	100 estimators, LR: 0.1
XGB	100 estimators, LR: 0.1, max. depth: 1

Table I.3
ILSVRC-2012 best parameters on Xception.

Model	Best parameters
MLP	MLP-2, LR: 0.0001
SVM	C: 10, γ: 0.0001, kernel: rbf
LR	C: 0.001, penalty: $\ell 2$
KNN	neighbours: 10
RFE	1000 estimators, entropy criterion
ADB	1000 estimators, LR: 0.1
GBC	100 estimators, LR: 0.1
XGB	100 estimators, LR: 0.1, max. depth: 1

References

Alaslani, M. G., & Elrefaei, L. A. (2018). Convolutional neural network-based feature extraction for IRIS recognition. *International Journal of Computer Science and Information Technology, 10*, 65–78.

Alipourfard, T., Arefi, H., & Mahmoudi, S. (2018). A novel deep learning framework by combination of subspace-based feature extraction and convolutional neural networks for hyperspectral images classification. In *IGARSS 2018 - 2018 IEEE international geoscience and remote sensing symposium* (pp. 4780–4783).

Ayinde, B. O., Inanc, T., & Zurada, J. M. (2019). *On correlation of features extracted by deep neural networks* arXiv:1901.10900.

Bodapati, J. D., & Veeranjaneyulu, N. (2019). Feature extraction and classification using deep convolutional neural networks. *Journal of Cyber Security and Mobility, 8*, 261–276.

Cao, X., Zhou, F., Xu, L., Meng, D., Xu, Z., & Paisley, J. (2018). Hyperspectral image classification with Markov random fields and a convolutional neural network. *IEEE Transactions on Image Processing, 27*(5), 2354–2367.

Chen, Y., Jiang, H., Li, C., Jia, X., & Ghamisi, P. (2016). Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing, 54*(10), 6232–6251.

Chen, Y., Lin, Z., Zhao, X., Wang, G., & Gu, Y. (2014). Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 7*(6), 2094–2107.

Chollet, F. (2016). *Xception: Deep learning with depthwise separable convolutions* arXiv:1610.02357.

Corchs, S., Fersini, E., & Gasparini, F. (2017). Ensemble learning on visual and textual data for social image emotion classification. *International Journal of Machine Learning and Cybernetics*.

Ding, L., Li, H., Hu, C., Zhang, W., & Wang, S. (2018). Alexnet feature extraction and multi-kernel learning for objectoriented classification. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-3*, 277–281.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh, & M. Titterington (Eds.), *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. In *Proceedings of Machine Learning Research: 9* (pp. 249–256). Chia Laguna Resort, Sardinia, Italy: PMLR.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. In S. Dasgupta, & D. McAllester (Eds.), *Proceedings of the 30th international conference on machine learning*. In *Proceedings of Machine Learning Research: Vol. 28* (pp. 1319–1327). Atlanta, Georgia, USA: PMLR.

HasanPour, S. H., Rouhani, M., Fayyaz, M., & Sabokrou, M. (2016). *Lets keep it simple, using simple architectures to outperform deeper and more complex architectures* arXiv:1608.06037.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016* (pp. 770–778). doi:10.1109/CVPR.2016.90.

Hertel, L., Barth, E., Käster, T., & Martinetz, T. (2017). *Deep convolutional neural networks as generic feature extractors* arXiv:1710.02286.

Hoffer, E., Hubara, I., & Soudry, D. (2018). Fix your classifier: The marginal value of training the last weight layer. In *International conference on learning representations*.

Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016). *Deep networks with stochastic depth* arXiv:1603.09382.

Jagusch, J.-B., Gonçalves, I., & Castelli, M. (2018). Neuroevolution under unimodal error landscapes: An exploration of the semantic learning machine algorithm. In *Proceedings of the genetic and evolutionary computation conference companion* (pp. 159–160). ACM.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., & Darrell, T. (2014). *Caffe: Convolutional architecture for fast feature embedding* arXiv:1408.5093.

Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images. *Technical Report*. Citeseer.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (pp. 1097–1105). Curran Associates, Inc..

Lapa, P., Gonçalves, I., Rundo, L., & Castelli, M. (2019). Enhancing classification performance of convolutional neural networks for prostate cancer detection on magnetic resonance images: A study with the semantic learning machine. In *Proceedings of the genetic and evolutionary computation conference companion*. ACM.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.

LeCun, Y., & Cortes, C. (2010). *MNIST handwritten digit database*.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., & Tu, Z. (2015). Deeply-supervised nets. In G. Lebanon, & S. V. N. Vishwanathan (Eds.), *Proceedings of the eighteenth international conference on artificial intelligence and statistics*. In *Proceedings of machine learning research: Vol. 38* (pp. 562–570). San Diego, California, USA: PMLR.

Li Fei-Fei, J. J., & Yeung, S. (2017). Stanford computer vision cs231n: Lecture 11 - detection and segmentation. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf. Accessed: 2018-01-02.

Lin, M., Chen, Q., & Yan, S. (2013). *Network in network* arXiv:1312.4400.

Ling, Z., Li, X., Zou, W., & Guo, S. (2018). Semi-supervised learning via convolutional neural network for hyperspectral image classification. In *2018 24th international conference on pattern recognition (ICPR)* (pp. 1–6).

Liu, B., Yu, X., Zhang, P., Tan, X., Yu, A., & Xue, Z. (2017). A semi-supervised convolutional neural network for hyperspectral image classification. *Remote Sensing Letters, 8*(9), 839–848.

Liu, X., Sun, Q., Meng, Y., Wang, C., & Fu, M. (2018). Feature extraction and classification of hyperspectral image based on 3d- convolution neural network. In *2018 ieee 7th data driven control and learning systems conference (DDCLS)* (pp. 918–922).

Lu, X., Yuan, Y., & Fang, J. (2017). JM-net and cluster-SVM for aerial scene classification. In *Proceedings of the twenty-sixth international joint conference on artificial intelligence, IJCAI-17* (pp. 2386–2392).

Ma, X., Fu, A., Wang, J., Wang, H., & Yin, B. (2018). Hyperspectral image classification based on deep deconvolution network with skip architecture. *IEEE Transactions on Geoscience and Remote Sensing, 56*(8), 4781–4791.

Minaee, S., Abdolrashidi, A., & Wang, Y. (2017). *An experimental study of deep convolutional features for iris recognition* arXiv:1702.01334.

Romero, A., Gatta, C., & Camps-Valls, G. (2016). Unsupervised deep feature extraction for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing, 54*(3), 1349–1362.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV), 115*(3), 211–252. doi:10.1007/s11263-015-0816-y.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in neural information processing systems* (pp. 3856–3866).

Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition* arXiv:1409.1556.

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. A. (2014). *Striving for simplicity: The all convolutional net* arXiv:1412.6806.

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). *Highway networks* arXiv:1505.00387.

Srivastava, S., Mukherjee, P., Lall, B., & Jaiswal, K. (2017). Object classification using ensemble of local and deep features. In *2017 ninth international conference on advances in pattern recognition (ICAPR)* (pp. 1–6). IEEE.

Stollenga, M. F., Masci, J., Gomez, F. J., & Schmidhuber, J. (2014). Deep networks with internal selective attention through feedback connections. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 27: Annual conference on neural information processing systems 2014, december 8–13 2014, Montreal, Quebec, Canada* (pp. 3545–3553).

Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *IEEE international conference on computer vision, ICCV 2017, Venice, Italy, October 22–29, 2017* (pp. 843–852). IEEE Computer Society. doi:10.1109/ICCV.2017.97.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In S. P. Singh, & S. Markovitch (Eds.), *Proceedings of the thirty-first AAAI conference on artificial intelligence, february 4–9, 2017, San Francisco, California, USA.* (pp. 4278–4284). AAAI Press.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., & Rabinovich, A. (2015). Going deeper with convolutions. In *IEEE conference on computer vision and pattern recognition, CVPR 2015, Boston, MA, USA, June 7–12, 2015* (pp. 1–9). IEEE Computer Society. doi:10.1109/CVPR.2015.7298594.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016* (pp. 2818–2826). doi:10.1109/CVPR.2016.308.

Szeliski, R. (2010). *Computer vision: Algorithms and applications*. Springer Science & Business Media.

Tang, Y. (2013). *Deep learning using linear support vector machines* arXiv:1306.0239.

Thomaz, R. L., Carneiro, P. C., & Patrocinio, A. C. (2017). Feature extraction using convolutional neural network for classifying breast density in mammographic images. In *Medical imaging 2017: Computer-aided diagnosis: Vol. 10134* (p. 101342M). International society for optics and photonics.

Valpola, H. (2015). From neural PCA to deep unsupervised learning. In *Advances in independent component analysis and learning machines* (pp. 143–171). Elsevier.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82.

Yang, J., Zhao, Y., & Chan, J. C. (2017). Learning and transferring deep joint spectral spatial features for hyperspectral classification. *IEEE Transactions on Geoscience and Remote Sensing, 55*(8), 4729–4742.

Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. In R. C. Wilson, E. R. Hancock, & W. A. P. Smith (Eds.), *Proceedings of the british machine vision conference 2016, BMVC 2016, York, UK, September 19–22, 2016*. BMVA Press.

Zhang, H., Meng, L., Wei, X., Tang, X., Tang, X., Wang, X., & Yao, W. (2019). *1d-convolutional capsule network for hyperspectral image classification* arXiv:1903.09834.