A comparative analysis between Convolutional Neural Networks and Support Vector Machines for image classification

Apostolos Kakampakos (03400133)
Data Science & Machine Learning
apostoloskakampakos@mail.ntua.gr

Christos Nikou (03400146)
Data Science & Machine Learning
christosnikou@mail.ntua.gr

Spyros Rigas (03400154)
Data Science & Machine Learning
spiridonrigas@mail.ntua.gr

With continuous advancements in technology that allow the development of complex models even on commercially available hardware, as well as an ever expanding range of applications, image recognition is one of the tasks that are currently at the forefront of machine learning research. This paper investigates the effectiveness of Convolutional Neural Networks in the sub-category of image classification via a comparative analysis between them and Support Vector Machines. A dataset of digital images corresponding to paintings is utilized for the purposes of this analysis with a total of 20 labels (artists). The samples are preprocessed and different types of features are subsequently extracted from them: HOGs and SIFT-related features for the SVM and RGB pixels for the CNN. Each type of classifier is then trained on the corresponding data and their evaluation on a common test set shows a clear superiority of basic-architecture CNNs, which achieve a 50% accuracy, compared to SVMs with Gaussian kernels which score 21% in the case of HOG features and 35% in the case of SIFT-related features. In order to highlight the margin for further improvement in the case of CNNs, transfer learning methods are employed to develop a more powerful CNN, which achieves an astonishing 81% accuracy on the classification of the images.

I. Introduction

While the theoretical foundations were set as early as in 1943 with the conception of the mathematical neuron [1], the paradigm shift in computer science towards what has come to be known as the field of machine learning was the result of a series of revolutionary technological advancements, such as the manufacturing of powerful Graphics Processing Units (GPUs). These developments enabled the massive implementation of machine learning tasks at an unprecedented rate, with image recognition standing out as one of the most prominent amongst them, due to its numerous applications in a variety of fields that are nowadays ranging from computer vision [2] to cancer detection [3].

One of the first milestones for image recognition [4] was the achievement of a record breaking accuracy on the classification of handwritten digits by training an upgraded version of Fukushima's [5] Neocognitron: a Convolutional Neural Network (CNN). While "conventional" classifiers such as Support Vector Machines (SVMs) [6, 7] or k-Nearest Neighbor (kNN) [8] have been reported to perform well on occasion for the task of image recognition, CNNs and upgraded or hybrid versions thereof remain, to this day, the uncontested champions. This is due to a number of reasons: their effectiveness on handling features even at the pixel level, their unique ability to reduce the dimensionality of the data, while maintaining a large fraction of their variance, as well as the increasing volume of available labeled images [9] to train on, to name but a few.

The aim of the present paper is to compare conventional classifiers to CNNs, as far as feature extraction and classification scores are concerned. The conventional classifiers of choice are SVMs with Gaussian (radial basis function - rbf) kernels and the studied problem is the classification of paintings. More specifically, each model is expected to receive input features extracted from digital images of the paintings and classify the painting by transmitting a number from 0 to 19, with a one-to-one correspondence to 20 different artists.

The remaining sections of this paper are structured as follows. In Section II, a brief review of studies related to image classification is presented, alongside a short discussion on the state-of-the-art for this machine learning task. Section III gives some basic information on the used dataset and provides a detailed analysis of the data preprocessing and feature extraction stages. After a short summary of the relevant theoretical concepts, the choice and development of the studied models are analyzed in Section IV. Subsequently, in Section V, the main results of the models' deployment are discussed and some important conclusions are drawn with regards to their relative performance. Finally, the key results are summarized in Section VI and an outlook on future work is presented.

II. Related Work

Image recognition is not, in fact, a machine learning task in itself, but rather an umbrella term encompassing a series of tasks, such as object detection [10], image segmentation [11], or shape recognition [12]. A direct consequence of this is that the state-of-the-art is somewhat ill-defined without additional context with regard to the specific task or the test dataset.

The task performed for the purposes of this paper falls into the sub-category of image classification, the current record accuracy score for which is 90.88% [13]. This score was achieved on ImageNet [9] by a hybrid known as CoAtNet, which combines the strengths of CNNs and structures known as Transformers [14]. Transformers are networks whose architecture is based solely on attention mechanisms and that

are currently the leading model for Natural Language Processing (NLP) tasks [15]. In fact, the second best accuracy score on ImageNet is 88.55% and was achieved by a "pure" Transformer [16]. Such cutting-edge models are far more complex compared to the CNN developed for the purposes of this paper's analysis, which is somewhat "simpler", in the sense that its architecture (more details in Section IV) closely resembles the architecture of VGGNet [17], which scored first on the ImageNet classification contest of 2014.

As far as the other classifier studied herein is concerned, SVMs have been successful in the classification of images with a small number of labels, like handwritten digits [18]. Nevertheless, when it comes to more complex images or larger numbers of labels, the use of SVMs is rarely reported in literature, unless they are combined with other powerful classifiers, namely the aforementioned CNNs [19]. This is partly due to the fact that, while CNNs achieve tremendous accuracies by training on features even at the pixel level, the training of SVMs (and other traditional classifiers) requires the extraction of more complex features from images, such as Histograms of Oriented Gradients (HOGs) [20] or features acquired from Scale-Invariant Feature Transforms (SIFT) [21]. As discussed in more detail in the following, these are also the features considered for the training of the SVMs in the experiments presented in this paper.

III. Data Preprocessing & Feature Extraction

The digital images of the paintings are retrieved from a Kaggle dataset by K. Nichol [22]. The dataset contains a total of 103251 unique paintings from 2319 artists, however only a subset of them is used for the training and evaluation of the developed models. More specifically, 20 artists are hand-picked so that there is a wide variety of genres and painting styles available in the data, but not in such a way that each artist is uniquely defined by them. For example, geometric shapes are abundant in Wassily Kandinsky's works, while Francisco Goya mainly depicts the human form, however the human form is a common theme in Henri de Toulouse-Lautrec's artworks as well. A total of 214 paintings[1] are chosen at random for each artist, thus creating a balanced dataset of 4280 samples. An example of one such sample is depicted in Fig. 1.

The data preprocessing stage consists of the following steps. As far as the sample labels are concerned, sklearn's [23] `LabelEncoder` is invoked in order to create a one-to-one correspondence between the artists' names and the integers from 0 to 19. When it comes to the samples themselves, at first, every image is loaded as a PyTorch [24] tensor object with dimensions (3, height, width), where 3 corresponds to the RGB channels. The image's RGB channels are then normalized by using ImageNet's mean and standard deviation, which are equal to $\mathbf{m} = (0.485, 0.456, 0.406)^\top$ and

[1]Among the 20 artists chosen, the artist with the smallest number of available paintings was Wassily Kandinsky, with a total of 214 paintings. A larger number of samples was not chosen for the other artists, in order to ensure a balanced dataset.

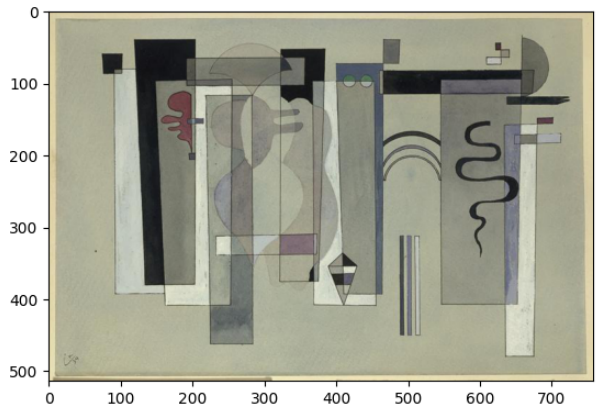

Fig. 1. Random sample chosen from the used dataset. The axes correspond to pixels and the depicted painting is *Green emptyness* by Wassily Kandinsky.

$\mathbf{s} = (0.229, 0.224, 0.225)^\top$, respectively. Finally, the image is resized to a fixed shape of 224×224 pixels, in order to ensure uniformity, since the dataset includes images of various resolutions. This resizing process is performed in a way that preserves the image's aspect ratio to avoid distortions: if the image's width is shorter than its height, the image is rotated[2] by 90 degrees so that the horizontal axis is always aligned with the largest dimension. The image is then resized uniformly so that the largest dimension becomes equal to 224 pixels and the empty area in the bottom of the 224×224 block is padded with zeros (or, equivalently, filled with black). It is worth noting that a common alternative is the cropping of a 224×224 sized square from the center (or any other area) of the image in order to be used as the sample, however the classification results obtained this way are not satisfactory, so this approach is dismissed. Fig. 2 depicts the sample from Fig. 1 once it has been processed using this procedure.

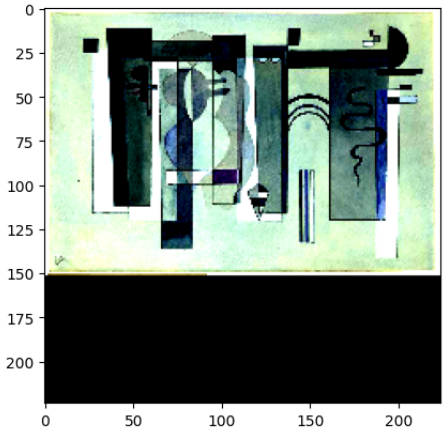

Fig. 2. The transformed version of the original sample depicted in Fig. 1.

Moving on to the feature extraction process, CNNs are known to perform outstandingly well even at the pixel level,

[2]While distortions could greatly impact the classification results, the same is not true for operations such as rotations or translations.

so the preprocessed samples as the one shown in Fig. 2 are ready to be inserted into the CNN's input with no further transformations. On the other hand, when it comes to traditional classifiers like SVMs, more complex features are required. For the purposes of this paper's analysis, two such features are extracted: HOGs and SIFT-related features, both of which are feature descriptors used for the purpose of object detection.

The extraction of HOGs requires the computation of the image's spatial gradients along its horizontal and vertical axis. The two perpendicular gradient vectors are then added into one vector which is defined by its magnitude and orientation on the plane. Afterwards, the image is divided into cells (small connected regions) and the gradient magnitude of each pixel in a cell is voted into different orientation bins according to the gradient's orientation. Adjacent cells are then grouped into blocks, which are in turn normalized by their L2-norm and concatenated to form a HOG. The HOGs that are extracted for this paper's purposes consider 9 discrete orientations on the plane, with cells consisting of 8×8 pixel areas and 2×2 cell blocks and the extraction is performed by skimage's [25] `feature.hog` method. A visual representation of a HOG can be seen in Fig. 3, where the HOG descriptor is extracted from the image shown in Figs. 1 and 2. It is evident that the outlines of the original image's shapes are clearly imprinted in the HOG descriptor. Note that the blank space at the bottom of the image is far from accidental: it simply reflects the fact that padded regions are not taken into consideration, as they are supposed to.

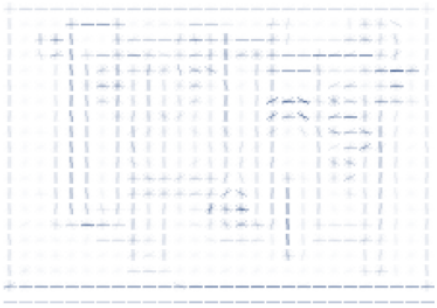

Fig. 3. The visual representation of the HOG descriptor for the sample depicted in Figs. 1 and 2.

Things are somewhat more complicated in the case of SIFT-related features, where a series of keypoints are detected across an image, as shown in Fig. 4. For each keypoint's neighborhood, a descriptor is constructed by extracting HOGs on 4×4 pixel blocks with 8 bins each. As a result, each keypoint corresponds to a 128-dimensional vector which serves as a feature. This implies that a different number of keypoints is extracted from different images, thus creating a large pool of features. A clustering algorithm is then applied in this large pool of 128-dimensional features, in order to create k clusters of descriptors. The image samples are then re-visited and a k-dimensional feature vector is assigned to each of them, with elements corresponding to the image's contribution to each cluster. The contribution to a cluster is defined as the number of the image descriptors that fall into the cluster, divided by the total number of descriptors for normalization purposes. This is effectively the equivalent of a Bag-of-Words approach [26] used in NLP problems. For the present analysis, the detection of keypoints is performed using OpenCV's `cv2` package [27] and k-Means is the algorithm chosen for the clustering process. The number of clusters is set equal to $k = 500$, so 500-dimensional feature vectors are extracted from each image.

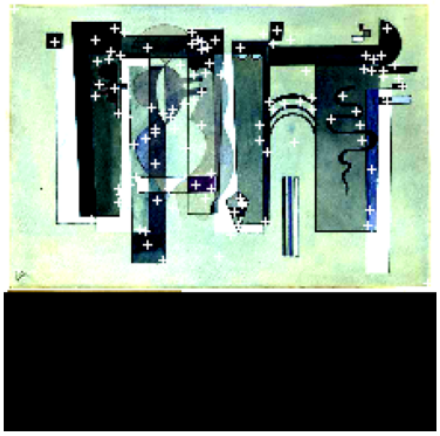

Fig. 4. The keypoints detected for the sample depicted in Fig. 1.

Before initiating the model training and evaluation, the dataset is split into training, validation and test data in a 60:25:15 ratio in all cases of features. It is worth noting that the calculation of the k cluster centroids for the SIFT-related features is performed using only the data reserved for the training, in order to avoid leakage into the validation/test set and thus influence the evaluation process.

IV. Studied Classifiers

Before presenting the main results from the experiments performed, it is worth summarizing the main aspects and features of the models developed for the classification. As far as the SVM is concerned, the reason why it is chosen over other "traditional" classifiers is because of its efficiency with high dimensional features, as is the case when dealing with features extracted from images.

In the binary problem case, given a series of n feature vectors $\mathbf{x}$ and labels $y \in \{1, -1\}$, the goal is to find a weight $[w_0, \mathbf{w}]$ such that the prediction given by $\text{sgn}\left(w_0 + \mathbf{w}^\top \phi(\mathbf{x})\right)$ is correct for most samples, where $\phi(\mathbf{x})$ is a mapping from the feature space spanned by $\mathbf{x}$. It can be proven that the problem of determining $[w_0, \mathbf{w}]$ is equivalent to the minimization of

$$\mathcal{L}(\alpha_1, \ldots, \alpha_n) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \quad (1)$$

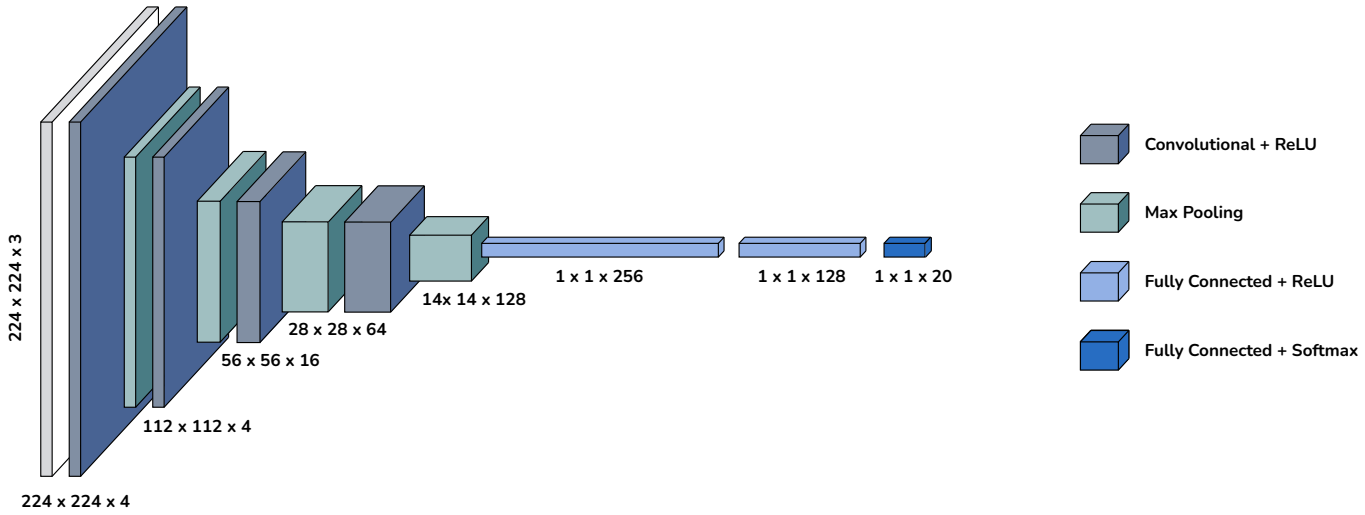

Fig. 5. CNN architecture consisting of 4 sequences of Convolutions, ReLU activations and Max Pooling operations, with an output that is inserted into a series of 3 FC layers with ReLU and Softmax activations.

with respect to the Lagrange multipliers $\alpha_1, \ldots, \alpha_n$, where

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \phi^\top\left(\mathbf{x}_i\right)\phi\left(\mathbf{x}_j\right) \tag{2}$$

is the so-called Kernel function. The Lagrange multipliers are subject to the condition $0 \leq \alpha_i \leq C$, $i = 1, \ldots, n$, where C corresponds to a misclassification penalty term. Given the Lagrange multipliers that minimize (1), the weight is calculated as

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \phi\left(\mathbf{x}_i\right), \qquad w_0 = \frac{1}{y_s} - \mathbf{w}^\top \phi\left(\mathbf{x}_s\right), \tag{3}$$

where s is the index of any sample with $\alpha_s \neq 0$, i.e. a support vector. Denoting the set of indices corresponding to the support vectors by SV, the classification prediction for a given feature vector $\mathbf{x}$ is given by the sign of

$$\sum_{i \in \mathrm{SV}} \alpha_i y_i K\left(\mathbf{x}_i, \mathbf{x}\right) + w_0. \tag{4}$$

In the multi-label problem studied here, a simple one-versus-one approach is implemented: for the total of $N = 20$ labels, $N\left(N-1\right)/2 = 190$ SVM classifiers are constructed by following the aforementioned steps, one for each pair of classes. Each classifier then gives a prediction based on Eq. (4) and a voting scheme is implemented to decide on the final classification of the sample by majority vote. For the purposes of the present analysis, this ensemble of SVM classifiers is developed by invoking sklearn's `SVC` class, using a Gaussian (rbf) Kernel, i.e.

$$K_{\mathrm{rbf}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2\right), \tag{5}$$

where $\gamma = \left(2\sigma^2\right)^{-1}$, σ being a Gaussian's standard deviation. The choice for the values of the γ and C parameters is discussed in the following Section.

When it comes to CNNs, the protagonist of this paper, the most important feature that differentiates them from multilayer perceptrons and renders them the perfect candidate for image classification is the combination of their convolutional and pooling layers. A convolutional layer calculates the convolution of its input with a lower-dimensional kernel, a procedure which effectively removes noise from the input data and regularizes them, creating a feature map. In the pooling layer, the feature map's neurons are combined into (in general overlapping) clusters which become equivalent to a single neuron. This equivalence depends on the nature of the pooling: max pooling is one of the most common pooling methods, where only the maximum value of each cluster of neurons is taken into account. Another characteristic of CNNs is that, even though any activation function can be used at the output of the convolutional or the pooling layers, the most common choice is the Rectified Linear Unit (ReLU), defined as

$$\mathrm{ReLU}\left(\mathbf{x}\right)_i = \max\left\{0, x_i\right\}. \tag{6}$$

Its advantage over activation functions like Softmax, where

$$\mathrm{Softmax}\left(\mathbf{x}\right)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}, \tag{7}$$

is that the problem of saturation is rectified, since its range is not the interval $[0, 1]$, but rather the interval $[0, +\infty)$.

For the development of the CNN, a new class inheriting from PyTorch's `nn.Module` is constructed from scratch. The network first receives a 224×224×3-dimensional vector at its input, corresponding to the 224×224 RGB image. A 4-channel 2D convolution with a ReLU activation is applied on this input, leading to a 224×224×4-dimensional output. The third dimension is equal to the number of convolutional filters, while the first two dimensions remain unchanged because the convolution kernel is chosen to be a square of length $K_c = 3$, with stride $S_c = 1$ and padding $P_c = 1$. Therefore, since the

input dimension is $L_{\text{in}} = 224$, the output dimension calculated from

$$L_{\text{out}} = \frac{L_{\text{in}} - K + 2P}{S} + 1 \tag{8}$$

is $L_{\text{out}} = 224$. Afterwards, a max-pooling operation with a square kernel of length $K_p = 2$ and stride $S_p = 2$ is performed, leading to a 112×112×4-dimensional vector, according to Eq. (8). This sequence of

$$\text{2D Convolution} \rightarrow \text{ReLU} \rightarrow \text{Max Pooling}$$

is repeated three times with identical K, S, P values, using 16, 64 and 128 filters, therefore leading to output vectors of size $(56, 56, 16)$, $(28, 28, 64)$ and $(14, 14, 128)$, respectively. The resulting 14×14×128-dimensional vector is then flattened and fed into three sequential Fully-Connected (FC) linear layers with output dimensions equal to 128, 64 and 20. The activation function for the first two FC layers is ReLU, while Softmax is chosen for the final layer, in order to perform the sample classification. This architecture is summarized in the sequence shown in Fig. 5.

V. Results & Discussion

The first part of the experimental procedure consists of the training and evaluation of the SVM ensemble using the HOG features. For this purpose, a triple-stage grid search is first performed for the values of the parameters γ and C, which are defined in Section IV and correspond to the problem's hyperparameters. During the first stage, 10 randomly chosen equidistant values in the range $[0, 100]$ are chosen for each hyperparameter, thus forming a 10×10 grid. For each hyperparameter pair, the SVM classifier is trained on the training set and evaluated (by accuracy) using the validation set. The hyperparameter pair for which the highest validation score is achieved becomes the center of a new 3x3 grid which is the initial state of the second stage, as depicted in Fig. 6.

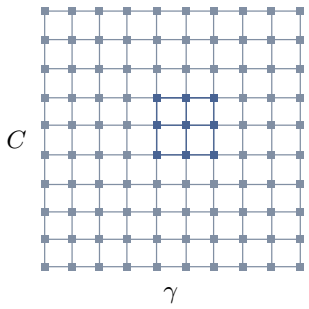

Fig. 6. A depiction of the grid at the end of the grid search's first-stage, during the process of hyperparameter tuning.

The resulting 3x3 grid is then transformed into a 5x5 grid, by considering the midpoints between the grid's nodes and the process of training and validation is repeated. The third stage is identical to the second stage and the final pair of the hyperparameters is utilized for the full training of the SVM. These values are $\gamma = 1.5$ and $C = 0.3$.

The actual training process is performed using the union of the training and validation sets and the total training time is calculated to be equal to 1101 seconds[3] (approximately 18 minutes), measured in CPU time. The SVM ensemble trained on HOG features is then evaluated on the data reserved for testing. As far as the evaluation is concerned, the final accuracy is not by itself a good measure to evaluate a multi-label classifier, which is why its corresponding macro and micro averaged versions are taken into account as well. The full classification report can be seen in Table I. For completeness, apart from the full classifier-level metrics, the Precision, Recall and F1-Scores are presented for the classification of each label separately, where the support column denotes the number of samples in each category.

	Precision	Recall	F1-Score	Support
0	0.12	0.09	0.11	32
1	0.00	0.00	0.00	32
2	0.14	0.62	0.22	32
3	0.02	0.03	0.03	32
4	0.00	0.00	0.00	32
5	0.10	0.09	0.10	32
6	0.29	0.41	0.34	32
7	0.61	0.84	0.71	32
8	0.34	0.78	0.47	32
9	0.00	0.00	0.00	32
10	0.75	0.19	0.30	32
11	0.00	0.00	0.00	32
12	0.06	0.12	0.08	32
13	0.18	0.19	0.18	32
14	0.00	0.00	0.00	32
15	0.10	0.19	0.13	32
16	0.33	0.28	0.31	32
17	1.00	0.03	0.06	32
18	0.00	0.00	0.00	32
19	0.31	0.34	0.33	32
Accuracy			**0.21**	642
Macro Averaged	0.22	0.21	**0.17**	642
Micro Averaged	0.22	0.21	**0.17**	642

Table I. Classification report for the SVM using HOG features.

The results indicate that the SVM ensemble developed using HOG features does not perform satisfactorily, achieving an accuracy score of 21%, with its macro and micro averaged versions assuming even lower values at 17%. From a statistical standpoint, as far as accuracy is concerned (i.e. ignoring the bias towards specific classifications described by the macro and micro averaged F1-Score) this result implies that roughly one out of five paintings is classified correctly. Another important observation is that, out of the total of 20 artists, there are 6 for whom there are no classifications assigned, even erroneous ones. Finally, the SVM ensemble appears to be biased towards the artist corresponding to label 2, since there is a significant deviation between their Precision and Recall scores.

[3] All training times are calculated on the standard machines used by Kaggle, since all code is written and executed on Kaggle environments. For a complete list of the machine specifications visit this link.

Further information can be extracted from the confusion matrix corresponding to the test samples' classification, which is given in Fig. 7. Apart from the correspondence between 0-19 labels and artists, the confusion matrix provides more information than Table I does, since it shows exactly how misclassifications occur. In other words, Table I can be constructed using the confusion matrix, although the reverse is not true.

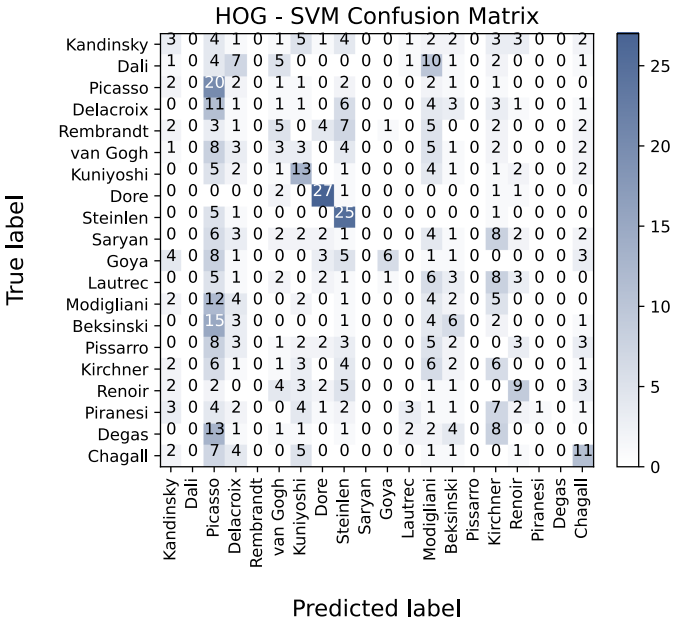

Fig. 7. The resulting confusion matrix from the classification by the SVM using HOG features.

It becomes evident that the artist towards whom the SVM is biased is Pablo Picasso, while Gustave Dore and Theophile Steinlen are the artists with the most correct classifications. The columns that are filled with zeros correspond to the entries of Table I that are zero for all measures. The corresponding artists are Salvador Dali, Rembrandt, Martiros Saryan, Henri de Toulouse-Lautrec, Camille Pissarro and Edgar Degas.

When it comes to the SVM ensemble trained on SIFT-related features, the same grid search scheme is performed. The optimal values for the hyperparameters are now found equal to $\gamma = 6.1$ and $C = 8.3$. Both of these values are higher compared to the ones calculated for HOG features and especially the resulting value for C (which is one order of magnitude higher) indicates that a hard-margin-type SVM is trained for the case of SIFT-related features when it comes to misclassification penalties. Using these values for the hyperparameters, the SVM ensemble is first trained on the union of the training and validation set and subsequently evaluated on the test set, achieving the scores shown in Table II.

While not in itself satisfactory, the SIFT-SVM shows a significantly improved performance compared to the HOG-SVM. First of all, the accuracy and the macro and micro averaged versions of the F1-Score are relatively closer, which means that the SIFT-SVM is not biased towards making specific types of misclassifications. Additionally, from a statistical perspective, the SIFT-SVM correctly classifies one out of three paintings. Finaly - and perhaps most importantly - the SIFT-SVM assigns paintings to all 20 artists, without ignoring any of them, and the bias towards specific painters (for example Picasso - label 2) appears reduced. Once again, the confusion matrix is also provided for a more complete overview of the classification results in Fig. 8.

	Precision	Recall	F1-Score	Support
0	0.42	0.34	0.38	32
1	0.40	0.53	0.45	32
2	0.32	0.53	0.40	32
3	0.26	0.34	0.30	32
4	0.41	0.53	0.47	32
5	0.33	0.28	0.31	32
6	0.32	0.31	0.32	32
7	0.47	0.56	0.51	32
8	0.44	0.75	0.56	32
9	0.17	0.09	0.12	32
10	0.42	0.34	0.38	32
11	0.11	0.06	0.08	32
12	0.38	0.19	0.25	32
13	0.34	0.59	0.43	32
14	0.33	0.16	0.21	32
15	0.10	0.06	0.08	32
16	0.30	0.31	0.31	32
17	0.40	0.38	0.39	32
18	0.22	0.18	0.20	32
19	0.48	0.41	0.44	32
Accuracy			**0.35**	642
Macro Averaged	0.33	0.35	**0.33**	642
Micro Averaged	0.33	0.35	**0.33**	642

Table II. Classification report for the SVM using SIFT-related features.

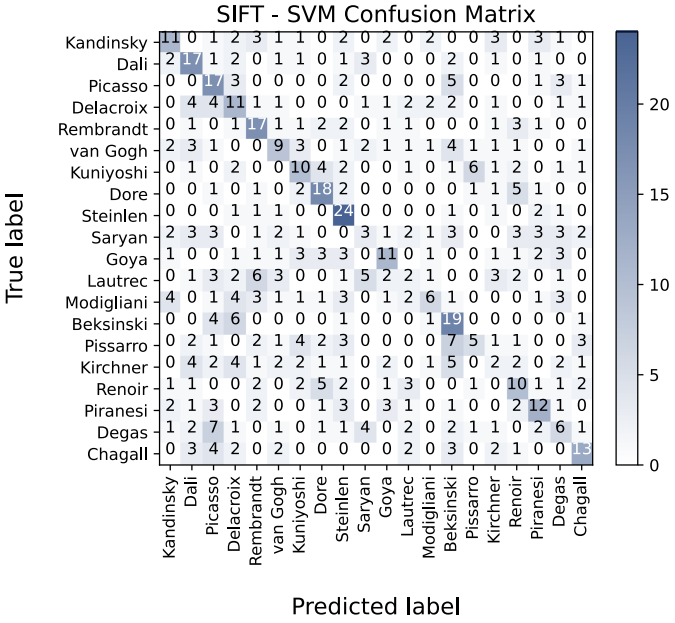

Fig. 8. The resulting confusion matrix from the classification by the SVM using SIFT-related features.

Indeed, the intensity of the diagonal is enough to conclude that the SVM constructed from SIFT-related features is a far better classifier compared to the one constructed from HOGs. This is not to say, however, that the SIFT-SVM shown here is an overall good classifier (only two correct classifications are performed for Lautrec and Kirchner); just that it is more accurate compared to the previously studied case.

Before concluding the discussion on the ensemble of SVM classifiers, it is important to note that the SIFT-SVM is not superior to the HOG-SVM in all aspects: while the training process itself takes only 13 seconds in CPU time for the case of the SIFT-SVM, since the size of the features is only 500 (equal to k), one should not ignore the duration of the feature extraction process, a part of which is the clustering of the SIFT descriptors extracted from the digital images. The process of loading the data and extracting the HOG features requires 351 seconds (approximately 6 minutes) in CPU time, while the process of loading the data and extracting the SIFT-related features requires a total of 6 hours, 18 minutes and 17 seconds in CPU time. This is the reason why exhaustive experiments are not performed for the parameter k, larger values of which would further increase the total extraction time[4].

With all the results acquired for the "traditional" classifier, the second part of the experimental procedure is associated with the training and evaluation of CNNs. Before initiating the training of the CNN model with the architecture described in Section IV, some additional features are implemented in order to make the training process faster and more efficient. First of all, the method of batch normalization [28] is applied during the network's training: for each batch, right before the application of each layer's nonlinear function, the activation vectors from hidden layers are normalized using the first and the second statistical moments (mean and variance) of the batch. This process is essentially a re-parametrization which effectively deals with the problem of coordinating updates across many layers, ensuring that slight changes at the higher-level weights do not lead to dramatic changes at the lower-level weights during backpropagation. As a result, the training of the network can be performed with generally higher learning rates and convergence is achieved faster [29].

Another feature integrated to increase training speed is the so-called early stopping. At the end of each training epoch, t, for a model trained using early stopping, the average loss, L_t, is calculated by evaluating the network on the validation data and compared to the average validation loss of the previous epoch, L_{t-1}. If $L_t < L_{t-1}$, a copy of the model is saved locally on the disk (checkpoint) and the training continues. However, if an increase in average validation loss is spotted, i.e. $L_t > L_{t-1}$, a countdown process is initiated. Given a patience threshold, p, if the average validation loss during any of the next p epochs is calculated to be lower than L_{t-1}, the countdown is reset, a new checkpoint is created and the training continues. If, on the other hand, $L_{t+k} > L_{t-1}$ for all $k = 1, 2, \ldots, p$, the training process is halted and the trained model returns to its last checkpoint. This process allows the choice of an arbitrarily large number of training epochs, since there is a guarantee that training will be halted once the minimum validation loss - given a threshold - is calculated. In addition, it ensures that overfitting will not significantly impact the model's generalizability. The early stopping algorithm implemented for the purposes of the present analysis is an adaptation of B. M. Sunde's class [30]. The patience threshold p is chosen equal to 10.

The final addition to the CNN model aiming to prevent overfitting is that of dropout layers [31], which are applied after the ReLU activations of the FC layers. The purpose they serve is that they prevent the development of strong dependencies between neurons (and by consequence overfitting) by randomly deactivating some neurons (setting their weights equal to zero), with probability $p\,(\text{dropout}) = 0.25$.

Taking everything into consideration, the CNN model is trained on the training set using batches of 32 samples. The optimizer used is Adam, with Cross Entropy as the loss function, since the problem at hand is a multi-label classification task. L2-regularization is also activated for the optimizer, with weight decay equal to 10^{-5}, in order to further reduce the rate of the network's overfitting. As far as the learning rate is concerned, several experiments showed that convergence is achieved for relatively small values, so a learning rate $\eta = 10^{-5}$ is chosen[5]. After training for 29 epochs, the early stopping mechanism is triggered and once the patience threshold is crossed the training is halted. The diagram shown in Fig. 9 depicts the training and validation loss curves for the CNN model's training, which required a total of 307 seconds (approximately 5 minutes) of CPU time.

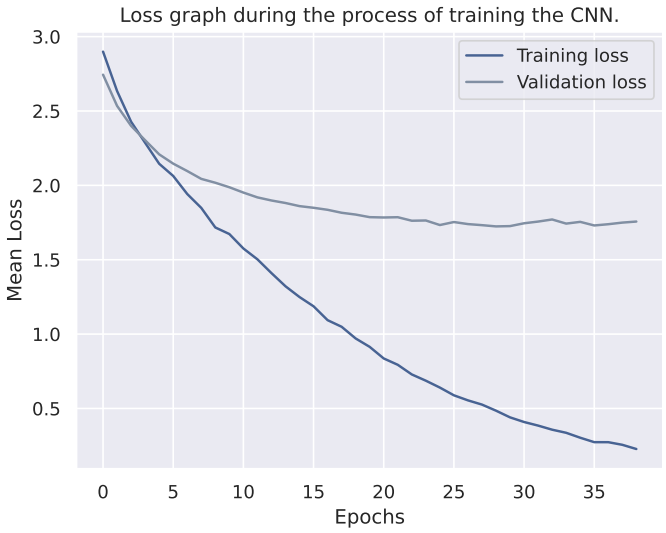

Fig. 9. Training (deep blue) and validation (soft blue) loss graphs for the CNN model's training.

[4]Note, however, that a test performed for $k = 200$ resulted in only slightly worse results in terms of accuracy.

[5]Without batch overfitting, the value of η would have to be even smaller.

An interesting remark is that the training process would require approximately 60-70 more epochs without batch normalization. The trained model is then evaluated on the test data, with the results shown in Table III. The corresponding confusion matrix is given in Fig. 10.

	Precision	Recall	F1-Score	Support
0	0.59	0.41	0.48	32
1	0.71	0.62	0.67	32
2	0.68	0.59	0.63	32
3	0.46	0.38	0.41	32
4	0.55	0.50	0.52	32
5	0.35	0.33	0.34	32
6	0.61	0.59	0.60	32
7	0.84	1.00	0.91	32
8	0.76	0.88	0.81	32
9	0.07	0.09	0.08	32
10	0.30	0.25	0.27	32
11	0.23	0.44	0.30	32
12	0.23	0.16	0.19	32
13	0.40	0.44	0.42	32
14	0.59	0.53	0.56	32
15	0.50	0.28	0.36	32
16	0.68	0.76	0.71	32
17	0.65	0.75	0.70	32
18	0.33	0.31	0.32	32
19	0.59	0.62	0.61	32
Accuracy			**0.50**	642
Macro Averaged	0.51	0.50	**0.50**	642
Micro Averaged	0.51	0.50	**0.50**	642

Table III. Classification report for the CNN.

The results obtained from the CNN model clearly surpass those obtained from the SIFT-SVM which, in turn, achieved a better accuracy compared to the HOG-SVM.

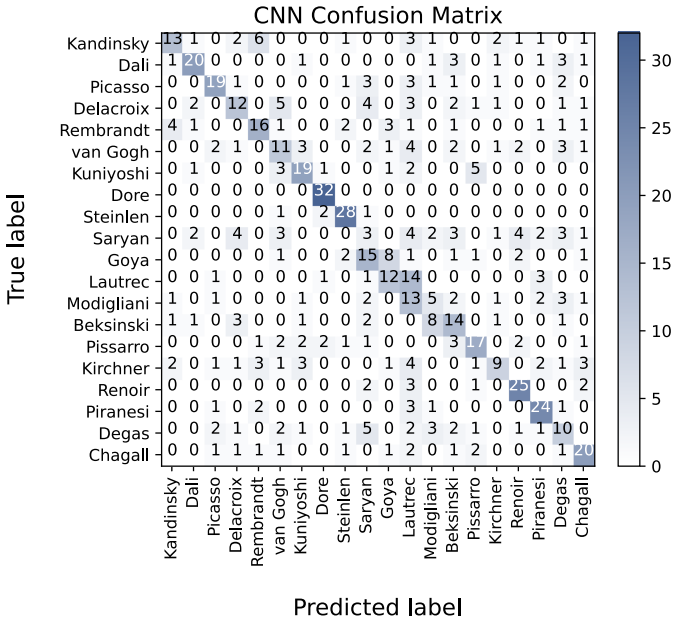

Fig. 10. The resulting confusion matrix from the classification by the CNN.

From a statistical standpoint, the CNN model classifies correctly one out of two paintings and shows no bias towards specific artists, since its accuracy score is in complete balance with the macro and micro averaged versions of the F1-Score. As can be seen from the confusion matrix, the color intensity is focused on the main diagonal, indicating the increased number of correct classifications compared to the results obtained from the SVM ensembles. There are still artists that the network does not correctly identify most of the time, like Saryan or Modigliani, both of whom the SIFT-SVM also had trouble identifying. Furthermore, the CNN seems to confuse Goya's and Lautrec's paintings, which can be attributed to the thematic similarity of the artists' paintings that was also commented in Section III. Despite these facts, the CNN's performance points to the superiority of CNNs compared to SVMs in all aspects: using the most basic of features (pixels) and dedicating less than 10 minutes of CPU time in total for the data loading, feature extraction and model training processes, the CNN model still managed to outperform - by a relatively big margin - the best-performing SIFT-SVM, the hyperparameters of which were carefully tuned.

The questions risen based on these results are the following: if a series of 4 convolutional and max pooling layers followed by 3 fully connected layers is enough to produce a classifier with 50% accuracy on a multi-label classification task, what is the margin for improvement by considering more complex architectures? Furthermore, how would a larger training dataset impact the classification results? Thanks to the method of transfer learning [32], these questions can be answered.

In transfer learning, a model pre-trained for a specific task on a large dataset is adapted in order to perform a similar task on a smaller dataset. This is usually done by removing a few of the pre-trained model's final layers (the number varies, depending on the similarity between the tasks and the datasets) and replacing them with new ones. The adapted model is then trained for the new task on the small dataset, either via freezing, where the pre-trained layers are frozen and only the replaced ones are trained, or by fine-tuning, where the entirety of the network is further trained on the new data.

For the purposes of the present analysis, the architecture of ResNet18 [33] provided by torchvision's [34] `models` module is utilized, using the method of fine-tuning. ResNet18 is a deep CNN which utilizes skip connections to jump over some layers and thus avoid the problem of vanishing gradients and mitigate the problem of accuracy saturation. After loading the ResNet18 which is pre-trained on ImageNet, the final FC layer is removed and replaced by two new FC layers with outputs equal to 256 and 20. Softmax is chosen as the activation of the final FC layer and the pre-trained model is fine-tuned using the same training and validation sets that were used for the training of the previous CNN. The training lasts for 33 epochs before early stopping is activated, which is equivalent to 168 seconds (roughly 3 minutes) in CPU time. The loss graphs of the ResNet's training are shown in Fig. 11. Before evaluating the ResNet on the test data, the fact that the minimum of the mean validation loss achieved is over 2 times lower compared

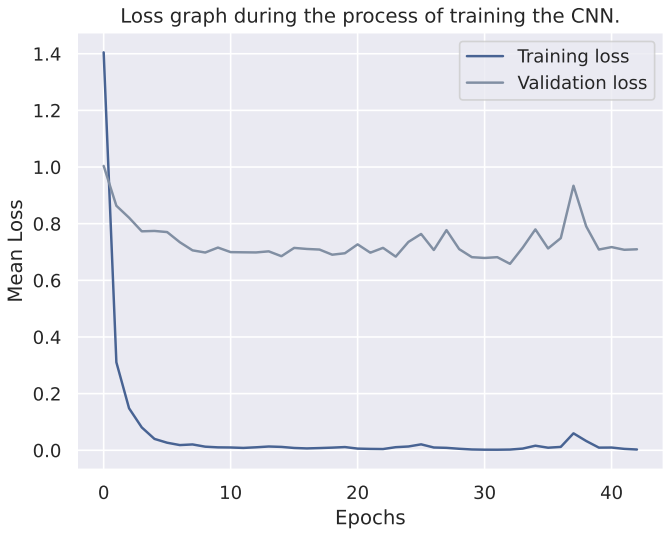

Fig. 11. Training (deep blue) and validation (soft blue) loss graphs for the ResNet18 model's training.

to the one achieved by the previous CNN is sufficient to deduce that the transfer learning of the ResNet will result in a better performance than the CNN's. To ascertain this, the ResNet is evaluated on the test set, with the results shown in Table IV.

	Precision	Recall	F1-Score	Support
0	0.85	0.91	0.88	32
1	0.85	0.91	0.88	32
2	0.83	0.75	0.79	32
3	0.85	0.72	0.78	32
4	0.93	0.78	0.85	32
5	0.74	0.70	0.72	32
6	0.73	0.84	0.78	32
7	1.00	1.00	1.00	32
8	0.97	0.97	0.97	32
9	0.52	0.53	0.52	32
10	0.76	0.81	0.79	32
11	0.81	0.91	0.85	32
12	0.67	0.81	0.73	32
13	0.82	0.88	0.85	32
14	0.87	0.84	0.86	32
15	0.78	0.56	0.65	32
16	0.77	0.82	0.79	32
17	0.88	0.94	0.91	32
18	0.81	0.78	0.79	32
19	0.90	0.81	0.85	32
Accuracy			**0.81**	642
Macro Averaged	0.82	0.81	**0.81**	642
Micro Averaged	0.82	0.81	**0.81**	642

Table IV. Classification report for the ResNet with transfer learning.

Indeed, transfer learning shows impressive results, with the ResNet classifier achieving an accuracy of 81%, completely consistent with the macro and micro averaged versions of the F1-Score. This 31% increase in accuracy compared to the previous CNN's highlights the importance of a large dataset to train from, as well as the endless possibilities of deep learning, where complex architectures can lead to elevated - sometimes unexplainable - performances. For completeness, the confusion matrix is also depicted in Fig. 12.

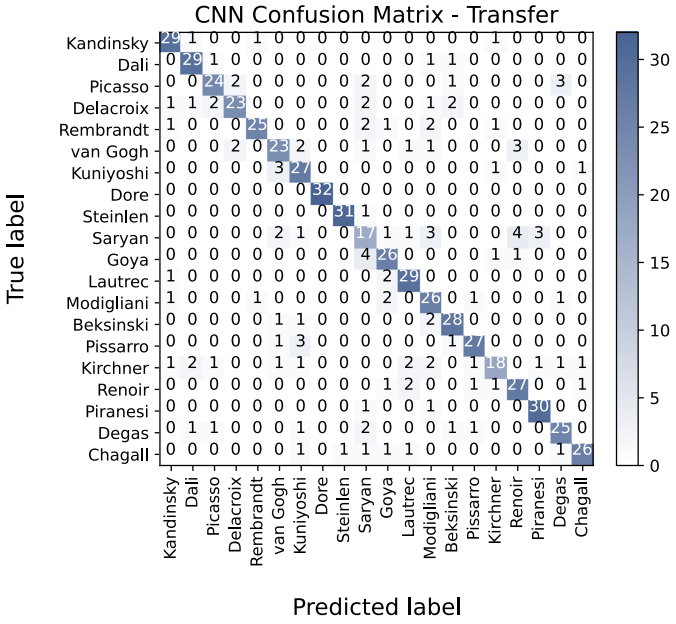

Fig. 12. The resulting confusion matrix from the classification by the ResNet.

The intensity of the confusion matrix is concentrated on the diagonal, with the number of misclassifications greatly reduced and artists like Modigliani, who were previously heavily misclassified, now having high Recall and Precision scores. Statistically, the ResNet classifies correctly roughly more than four out of five paintings.

VI. Outlook

To reiterate, with a dataset consisting of digital images of paintings, the problem of 20-label classification was addressed using traditional (SVM) classifiers, as well as CNNs. The SIFT-related features extracted to train and evaluate the SVM ensembles proved to be better candidates compared to HOGs, since the classification accuracy was 35% in the former case, compared to 21% in the latter. However, the CNN outperformed both SVM versions, scoring a 50% accuracy by training on the most trivial of features: pixels. The elevated performance of the CNNs was also accompanied by relatively short data extraction and training times, especially compared to the SIFT-SVMs. Using transfer learning to investigate the margin of improvement for CNNs with more complex structures, the classification accuracy reached an impressive 81%, proving that the 50% score achieved by the basic-architecture CNN was not close to saturation.

Convolutional neural networks undoubtedly proved to be superior to SVMs for the task of image classification, in all aspects. Despite that, there may still be room for improvement in the case of SVMs. On the level of feature extraction, different values of k (number of clusters) can be explored,

something that was not possible for the present analysis due to time limitations. Furthermore, a one-vs-all approach can be examined for the construction of the SVM ensemble, even though it is highly likely that it will not lead to better results (though it will definitely lead to lower training times).

When it comes to the future aspects of the deep learning models and image classification, more complex CNN architectures are definitely one of the directions to investigate. In addition, extracting features beyond the pixel level is an interesting idea, as the results would provide information on whether convolutions themselves are powerful enough to extract all the relevant information from the pixels, or if their performance can be boosted by more careful feature extraction. Last but not least, transformer networks are already dominating the NLP landscape and have gained a lot of attention in image recognition as well. Based on this, combinations of transformers with CNNs, like the current ImageNet record-holder, are certainly more than promising.

REFERENCES

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biophysics, vol. 5, no. 4, pp. 115–133, 1943, DOI: 10.1007/BF02478259.

[2] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, DOI: 10.1109/CVPR.2016.90.

[3] D. Mzurikwao et al., "Towards image-based cancer cell lines authentication using deep neural networks," Sci. Rep., vol. 10, no. 1, p. 19857, 2020, DOI: 10.1038/s41598-020-76670-6.

[4] Y. LeCun et al., "Backpropagation Applied to Handwritten Zip Code Recognition," Neural Computation, vol. 1, no. 4, pp. 541-551, Dec. 1989, DOI: 10.1162/neco.1989.1.4.541.

[5] K. Fukushima, "Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biol. Cybern., vol. 36, no. 4, pp. 193–202, 1980, DOI: 10.1007/BF00344251.

[6] C. H. Qian, H. Q. Qiang, and S. R. Gong, "An Image Classification Algorithm Based on SVM," Appl. Mech. Mater., vol. 738–739, pp. 542–545, 2015, DOI: 10.4028/www.scientific.net/AMM.738-739.542.

[7] X. Sun, L. Liu, H. Wang, W. Song and J. Lu, "Image classification via support vector machine," 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), 2015, pp. 485-489, DOI: 10.1109/ICCSNT.2015.7490795.

[8] S. Agustin and R. Dijaya, "Beef image classification using K-nearest neighbor algorithm for identification quality and freshness," J. Phys. Conf. Ser., vol. 1179, no. 1, p. 012184, 2019, DOI: 10.1088/1742-6596/1179/1/012184.

[9] J. Deng, W. Dong, R. Socher, L. -J. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248-255, DOI: 10.1109/CVPR.2009.5206848.

[10] Z. Shi, "Object Detection Models and Research Directions" 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE), 2021, pp. 546-550, DOI: 10.1109/ICCECE51280.2021.9342049.

[11] Y. Song and H. Yan, "Image Segmentation Techniques Overview," 2017 Asia Modelling Symposium (AMS), 2017, pp. 103-107, DOI: 10.1109/AMS.2017.24.

[12] H. Su, S. Maji, E. Kalogerakis and E. Learned-Miller, "Multi-view Convolutional Neural Networks for 3D Shape Recognition," 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 945-953, DOI: 10.1109/ICCV.2015.114.

[13] Z. Dai, H. Liu, Q. V. Le and M. Tan, "CoAtNet: Marrying Convolution and Attention for All Data Sizes," arXiv [cs.CV], 2021, Online: arXiv:2106.04803.

[14] A. Vaswani et al., "Attention is All you Need," Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017, Online: Here.

[15] A. Gillioz, J. Casas, E. Mugellini and O. A. Khaled, "Overview of the Transformer-based Models for NLP Tasks," 2020 15th Conference on Computer Science and Information Systems (FedCSIS), 2020, pp. 179-183, DOI: 10.15439/2020F20.

[16] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv [cs.CV], 2020, Online: arXiv:2010.11929.

[17] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv [cs.CV], 2015, Online: arXiv:1409.1556v6.

[18] E. Tuba and N. Bacanin, "An algorithm for handwritten digit recognition using projection histograms and SVM classifier," 23rd Telecommunications Forum Telfor (TELFOR), 2015, pp. 464-467, DOI: 10.1109/TELFOR.2015.7377507.

[19] S. Y. Chaganti et al., "Image Classification using SVM and CNN," 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), 2020, pp. 1-5, DOI: 10.1109/ICCSEA49143.2020.9132851.

[20] W. Zhou, S. Gao, L. Zhang and X. Lou, "Histogram of Oriented Gradients Feature Extraction From Raw Bayer Pattern Images," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 5, 2020, pp. 946-950, DOI: 10.1109/TCSII.2020.2980557.

[21] Q. Li and X. Wang, "Image Classification Based on SIFT and SVM," 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), 2018, pp. 762-765, DOI: 10.1109/ICIS.2018.8466432.

[22] K. Nichol, "Painter by Numbers," Retrieved February 05, 2022 from https://www.kaggle.com/c/painter-by-numbers/overview.

[23] F. Pedregosa, et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, 2011, pp. 2825–2830.

[24] A. Paszke, et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Advances in Neural Information Processing Systems 32, 2019, pp. 8024–8035, Online: Here.

[25] S. Van der Walt, et al., "scikit-image: image processing in Python," PeerJ, 2014.

[26] W. A. Qader, M. M. Ameen and B. I. Ahmed, "An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges," 2019 International Engineering Conference (IEC), 2019, pp. 200-204, DOI: 10.1109/IEC47844.2019.8950616.

[27] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

[28] V. Thakkar, S. Tewary and C. Chakraborty, "Batch Normalization in Convolutional Neural Networks — A comparative study with CIFAR-10 data," 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT), 2018, pp. 1-5, DOI: 10.1109/EAIT.2018.8470438.

[29] Z. Fu, S. Li, X. Li, B. Dan and X. Wang, "Influence of Batch Normalization on Convolutional Neural Networks in HRRP Target Recognition," 2019 International Applied Computational Electromagnetics Society Symposium - China (ACES), 2019, pp. 1-2, DOI: 10.23919/ACES48530.2019.9060588.

[30] B. M. Sunde, "Early Stopping for PyTorch," GitHub, 2013. Online: Here.

[31] K. Sanjar, A. Rehman, A. Paul and K. JeongHong, "Weight Dropout for Preventing Neural Networks from Overfitting," 8th International Conference on Orange Technology (ICOT), 2020, pp. 1-4, DOI: 10.1109/ICOT51877.2020.9468799.

[32] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks?," arXiv [cs.CV], 2014, Online: arXiv:1411.1792.

[33] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, DOI: 10.1109/CVPR.2016.90.

[34] S. Marcel and R. Yann, "Torchvision the Machine-Vision Package of Torch," Proceedings of the 18th ACM International Conference on Multimedia, 2010, pp. 1485–1488, DOI: 10.1145/1873951.1874254.

[35] J. D. Hunter, "Matplotlib: A 2D graphics environment," Computing in Science & Engineering, vol. 9, no. 3, 2007, pp. 90–95.

[36] P. Umesh, "Image Processing in Python," CSI Communications, 2012.