International Journal of Computers and Applications

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/tjca20

# Customised ResNet architecture for subtle color classification

Jari Isohanni

**To cite this article:** Jari Isohanni (2025) Customised ResNet architecture for subtle color classification, International Journal of Computers and Applications, 47:4, 341-355, DOI: 10.1080/1206212X.2025.2465727

**To link to this article:** https://doi.org/10.1080/1206212X.2025.2465727



Published online: 19 Feb 2025.

INTERNATIONAL JOURNAL OF COMPUTERS AND APPLICATIONS
2025, VOL. 47, NO. 4, 341–355
https://doi.org/10.1080/1206212X.2025.2465727

Taylor & Francis
Taylor & Francis Group



# Customised ResNet architecture for subtle color classification

Jari Isohanni

Digital Economy, University of Vaasa, Vaasa, Finland

**ABSTRACT**

This study addresses the challenge of recognizing subtle color differences, a problem critical to applications in fields such as healthcare, food production, and civil engineering. Specially research focusses on printed colors. The research evaluates multiple ResNet architectures, including ResNet-18, ResNet-34, and ResNet-50, to identify the most effective model for this task. Modifications to the ResNet-34 architecture are proposed, such as replacing average pooling with global max pooling and introducing max pooling layers within residual blocks, to enhance feature extraction and classification accuracy. The models were validated using a *K*-fold cross-validation, which confirms the effectiveness of the proposed approaches. The findings demonstrate the potential of these modifications to achieve high classification accuracy, showcasing their adaptability to real world scenarios. However, limitations such as the use of a specific dataset and the type of printer highlight the need for further research to generalize the approach across diverse datasets and conditions.

**ARTICLE HISTORY**
Received 24 May 2024
Accepted 5 February 2025

**KEYWORDS**
Machine vision; color difference; printed colors; convolutional neural networks (CNN); ResNet; max pooling

## 1. Introduction

ResNet (Residual neural network) by He et al. is a deep learning architecture for image classification and computer vision tasks. Unlike traditional convolutional neural networks (CNNs), ResNet uses skip connections. These connections allow the neural network to learn efficiently through residual mappings. Before ResNet was published, deep networks had mechanisms which attempted to directly approximate the underlying mapping. ResNet architecture presented learning by residuals, the difference between the input and output of a given layer [1–3].

ResNet architecture emerged as an advance in deep learning, addressing critical challenges associated with training very deep neural networks. By introducing skip connections, ResNet mitigates the vanishing gradient problem, enabling effective training of networks with hundreds or even thousands of layers. This not only improves the flow of gradients during backpropagation but also enhances the reuse of features across layers, leading to better representation learning [4, 5]. ResNet's ability to capture fine-grained details and subtle variations in data makes it a good choice for numerous computer vision applications, including image classification, object detection, and feature extraction. During the year 2024 over 41 000 articles mentioning ResNet were published, according to Google Scholar search.

ResNet architecture adds shortcuts (skip connections) every two layers. This enables the layers of residual networks to learn from residual mappings, this learning helps in representing identity mappings, and finally prevents networks from degradation as depth increases. The novelty of adding shortcuts every two layers is crucial, as previous research has shown that using skip connections on every or every third layer does not achieve the same performance [6].

The innovation of ResNet has sparked research around its approach, and there have been interest in modifying, extending, and adapting the ResNet architecture. Targ et al. presented an approach that adds convolutional layers and data paths to each layer [7]. Wide Residual Networks is a version of ResNet invented by Zagoruyko and Komodakis in their research where they showed the importance of residual blocks. Wide residual networks can outperform deep networks in some use cases, and their computational cost is lower [8]. Li and He explained how identity shortcut connections solve gradient fading problems and proposed adjustable shortcut connections. The authors stated that identity mappings are not reasonable to be adopted for all layer parameters [9]. HS-ResNet, an approach proposed by Yuam et al., implements a plug-and-play block that can be added to existing networks. HS-ResNet implements hierarchical split and concatenate connections within one single residual block [10]. Targ et al. came up in their research with ResNet in ResNet (RiR), RiR has a deep dual-stream architecture that generalizes ResNet and optimizes ResNet performance [10]. Stable ResNet by Hayou et al. addressed the problem of an exploding gradient which can occur if ResNet becomes very deep. Their approach looks to stabilize the gradient [11]. Another version, CO-ResNet was proposed by Bharati et al. Their model was optimized to detect COVID-19 in X-ray images. The authors mostly focused on hyperparameter tuning [12].

The popularity and good performance of the ResNet architecture have resulted in not only modified approaches, but also different depth variations of the ResNet. These variations are named based on how deep they are; for example, ResNet-18 has 18 neural network layers. Different versions of ResNet are, for example, ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-110, ResNet-152, ResNet-164, and ResNet-1202. These variations and their performance are evaluated in the first experiments of this research.

ResNet and its different versions have proven that the ResNet capabilities are useful in image classification tasks in many different use cases [13–19]. In one recent research, ResNet was used to classify different printed colors, in this use case ResNet achieved a high accuracy of 95% even when the CMYK color intensity was only 5% in some of the color channels.

The recognition of small color differences is not a commonly studied topic, but it is closely related to the development of functional inks. These inks are used in the packaging industry and can

**CONTACT** Jari Isohanni x2603813@student.uwasa.fi, jari.isohanni@gmail.com Digital Economy, University of Vaasa, Wolffintie 34, Vaasa 65200, Finland

be used as sensors for environmental values such as temperature or humidity. This is possible as functional inks change color according to their exposure to the variables [20].

In this research, ResNet variations are explored to see which of them can best classify subtle color differences. The best ResNet variation is selected for further fine-tuning to find out if it can be modified for optimized accuracy. Fine-tuning includes changing the architecture based on findings of the past research.

This article makes contributions to the optimization of ResNet architectures for addressing the challenge of detecting subtle color differences in printed sources, a task with significant implications in various applications. The study examines multiple ResNet models, including ResNet-18, ResNet-34, and ResNet-50, to identify a baseline architecture suitable for fine-grained color classification. Based on this foundation, modifications are proposed to the ResNet-34 architecture, focusing on adjustments to pooling strategies and feature extraction within residual blocks. The approach presented highlights the adaptability of ResNet architectures to subtle color classification tasks and underscores the potential of targeted architectural modifications to improve their performance. The findings can be used in various applications in sectors such as civil engineering, food production, and healthcare care, where precise color differentiation is needed.

This research is structured as follows. Section 2. contains an overview of the past research. Section 3. describes the methods and materials used, including the dataset, ResNet architecture, and other methods related to this research Section 4. describes the experiments used and their results. Section 5. discusses the results and provides information for future research. Section 6. goes trough the conclusions of the research.

## 2. Related past research

ResNet has been used to solve various problems in the past; some of the previous research uses different ResNet depth variations and applies them to specific use cases. Some past research adapts and modifies ResNet in various ways. This section summarizes some of the most relevant work done around ResNet in the past.

Sarwinda et al. used ResNet in the detection of colorectal cancer, experimenting with ResNet-50 and ResNet-18. All of their images were pre-processed with the contrast enhancement CLAHE method before making the dataset. In this research ResNet-18 reached an accuracy of 85% and ResNet-50 88% [14].

Subrataio et al. tuned the ResNet-101 hyperparameters to create a CO-ResNet model. This model was used to classify COVID-19 and pneumonia X-ray images. Their detection rate was 98.74% in cases of COVID-19, 92.08% and 91.32% in cases of normal and pneumonia. They also tested ResNet-152 which did not reach 90% accuracy [12].

Almoosawi and Khudeyer based their approach on ResNet-34 when they researched the identification of diabetic retinopathy. In their image dataset, the differences are quite small as in the research presented in this research. The F1-score of their solution was 93.2%. The F1-score is a combination of precision and recall, providing a balance between the two [21].

Yu et al. research recognition of early-stage breast cancer using the ResNet-50 architecture. In this research, SCDA (Scaling and Contrast Limited Adaptive Histogram Equalisation Data Augmentation) was used as a data augmentation tool. The model and approach used in this research reached 95.74% [16].

Gao et al. used ResNet-34 architecture and transfer learning as they had only a small amount of samples. ResNet-34 classified some classes with accuracy 100%, and even in the most challenging class, the leaf knot, the accuracy was 97%. The Leaf knot classification is closely related to the small difference classification problem presented in this research [17].

Hu et al. combined two ResNet-50 and one ResNet-34 architectures into a multidimensional feature compensation residual neural network. Each dimension was responsible for certain classifications related to crop diseases. As the last layer, authors had a compensation layer which used a compensation algorithm to determine final recognition result. Their approach achieved 85 % accuracy, which was better than other approaches with their dataset [22].

Al-Haija and Adebanjo worked on their research with the breast cancer dataset. With the ResNet-50 architecture and transfer learning, they achieved very high 99% accuracy [23].

ResNet-50 has shown its accuracy in disease identification in the past. Potato plant leaf disease identification was done by Shaheed et al. In this research ResNet reached 99.12% accuracy in only under 30 epochs. Zhang et al. performed another study that used ResNet-50 in disease identification. The researchers also used the coordinate attention module (CA) and the weight-adaptive multiscale feature fusion (WAMFF) and were able to achieve accuracy of 98.32 %. In addition, Li and Rai researched the identification of diseases in apples; however, they proved that shallow ResNet-18 outperformed ResNet-34 [24–26].

The summary of the previous research is shown in the following table.

| Research title | Authors | Findings |
|---|---|---|
| Sarwinda et al. | Deep learning in image classification using residual network (ResNet) variants for detection of colorectal cancer | Comparison of the ResNet-18 and ResNet-50 |
| Subrataio et al. | CO-ResNet: Optimized ResNet model for COVID-19 diagnosis from X-ray images | Hyperparameter tuning to increase performance of the ResNet-101 |
| Almoosawi and Khudeyer | ResNet-34/DR: a residual convolutional neural network for the diagnosis of diabetic retinopathy | Using of preprocessing to increase performance of the ResNet |
| Yu et al. | ResNet-SCDA-50 for breast abnormality classification | Using contrast enhancement to improve ResNet performance in abnormality classification |
| Gao et al. | A Transfer Residual Neural Network Based on ResNet-34 for Detection of Wood Knot Defects | ResNet was used to identify differences in wood, some of the changes were quite subtle |
| Hu et al. | MDFC–ResNet: An Agricultural IoT System to Accurately Recognize Crop Diseases | Using of ResNet to identify small differences in images |
| Al-Haija and Adebanjo | Breast Cancer Diagnosis in Histopathological Images Using ResNet-50 Convolutional Neural Network | Identifying features from images |
| Zhang et al. | Classification and Identification of Apple Leaf Diseases and Insect Pests Based on Improved ResNet-50 Model | Improving ResNet-50 with coordinate attention (CA) module and weight-adaptive multi-scale feature fusion |
| Shaheed et al | EfficientRMT-Net – An Efficient ResNet-50 and Vision Transformers Approach for Classifying Potato Plant Leaf Diseases | Integration of Vision Transformer (ViT) and ResNet-50 architectures |
| Li and Rai | Apple leaf disease identification and classification using resnet models | Comparison of ResNet-18 and ResNet-34 in lead disease identification |

Previously ResNet has been used in various use cases, and the mentioned research are closely related to the use case presented in this research. The similarity comes from the problem of recognizing subtle differences in images. Previously, ResNet and related architectures have been shown to perform well when small differences in images are observed or located. Most research in the past has used the standard version of the ResNet, the most popular being 50-layer and 34-layer versions. These previous researches have mainly focused on improving the performance of ResNet via pre-processing or other methods that do not directly modify the architecture of ResNet. But there are also examples of modifications that make the architecture work better in specific use cases. In addition, there is some research showing that increasing the depth of the ResNet architecture does not correlate with accuracy. As ResNet achieves high accuracy in multiple of the presented cases, choosing the best optimizer or hyperparameter tuning has not been a very popular topic in relation to ResNet. Another lesson that can be learnt from the previous research is that using enough correct data augmentation helps when the generic and more accurate model is the objective.

The main gap in previous research is that ResNet (or any other CNN architecture) has not been used to classify subtle color differences. This research explores how ResNet can be used in this use-case and if better architecture can be developed with small changes. This article proposes two modified ResNet-34 architectures tailored to the subtle color difference classification in printed sources. The first modification replaces the average pooling layer with global maximum pooling before the fully connected layer, enhancing feature extraction by focusing on the most significant features across the input space. The second modification adds a maximum pooling layer after each convolutional operation within residual blocks, improving the network's ability to capture subtle differences by emphasizing local maxima. These adjustments are supported by additional techniques such as gradient centralization and the use of $K$-fold cross-validation for robust performance evaluation. These proposed approaches overcome identified gaps by improving model accuracy, reducing overfitting, and tailoring ResNet-34 for subtle color classification.

## 3. Methods and materials

### 3.1. The dataset

The dataset [27] used in this research contains images that have been acquired using normal smartphones in various real-life environments. The images contain a special QR-Code as a carrier of color information (Figure 1). Before any operation, the source images are run through an auto-leveling process. This process is a technique commonly used in image processing to automatically adjust the contrast of the image. Auto-leveling aims to spread out the brightness levels across the entire dynamic range available. The purpose of image auto-levelling is to enhance presentation of colors by automatically adjusting the brightness, contrast, and color balance of an image to achieve a more natural and evenly distributed tonal range. The auto-levelling process starts with the identification of the minimum and maximum intensity values within the image. After determining the minimum and maximum intensity values, each pixel's intensity in the image is mapped to a new value based on a linear transformation. Pixels with intensities below the minimum value are assigned 0 (black), while pixels with intensities above the maximum value are assigned 255 (white). The intensities between are linearly scaled to cover the entire dynamic range. The mapping process is done for all RGB-channels [28]. An example of auto-levelling can be seen in Figure 1. where image 1 is before and image 2 after auto-levelling. After the auto-levelling the image goes through a Gaussian blurring. Blurring reduces noise in an image by averaging the intensity values of neighboring pixels, thereby smoothing out abrupt variations caused by random noise or, as in the use case presented, printer patterns. Noise typically appears as sharp, high-frequency intensity fluctuations that stand out from the underlying signal or pattern in the image. Blurring applies a low-pass filter, which suppresses these high-frequency components while retaining the overall structure of the image.

The actual color information used is placed at two specific locations on the QR-Code. These locations and their cornerpoints are calculated as part of the QR-Code decoding process. The locations and their cornerpoints are used to correct area skew and orientation, resulting in square color areas. These color areas are then extracted into separate images for pre-processing and forming of the final dataset. The extraction process extracts 80% from the color area to ensure that only the needed area is left. In this way, the possible distortion and skew that is left will not impact the process.

Each of the source images originally contains two different areas (Figure 1). The first area is a rectangle with paper-white (green rectangle), e.g. no color printed, and the second one has some color printed on it (yellow rectangle). Printed color is the one CNN model is expected to classify. In the process, first an average RGB value for the paper-white area is calculated, this is done with the following formula:

$$\bar{C}(c) = \frac{1}{M}\sum_{x=1}^{W}\sum_{y=1}^{H} I(r,g,b) \tag{1}$$

where $I$ is the source image and $R$, $G$, and $B$ represent individual integer values in each RGB channel at pixel location $(x, y)$. $W$ and $H$ are the width and height of the image and $M$ is the total number of pixels in the image.

To form the final image, used as the dataset image, the average color of the white area is subtracted from the color area. This creates a difference image which contains information about how much color area differs from no color area. The difference image is calculated with the following formula:

$$I_{\text{difference}}(r,g,b) = |I_{\text{original}}(r,g,b) - C(r,g,b)| \tag{2}$$

where $I_{\text{difference}}$ is the final image and $I_{\text{original}}$ is the color area image (Figure 1(b)). Constant $C$ is the average RGB color value in the white area Figure 1(a).

In Figure 2 process of forming the difference image is shown with a few examples. In Figure 2 row (a) contains sample images of average paper-white color, row (b) contains the actual color area, and row (c) is the final dataset image. The columns in the figure represent different colors printed on paper 10% yellow, 10% cyan, 10% magenta, 5% yellow, 5% cyan, 5% magenta. In the final difference images (Figure 2 row (c)), it can be seen that the differences between the samples are very small. This makes it difficult for the convolutional neural network (CNN) to classify images correctly.

The whole process of making the dataset is illustrated in Figure 3.

The dataset is augmented with the following options, data augmentation is one option to enlarge the dataset and can provide better accuracy [29]. Images are rotated in angles, 90, 180, and 270, then images are flipped vertically and horizontally, and the same rotations are done for flipped version of images. Also, the 0.2 zoom option is used to make more images using data augmentation.

In total, the dataset contains 7855 RGB images. Each image is stored with 24-bit depth, which means 8 bits per each channel. The images are split into train (80%) and validation (20%) sets so that the train set contains 6284 images belonging to 11 classes and the validation set contains 1571 images belonging to 11 classes (Table 1).

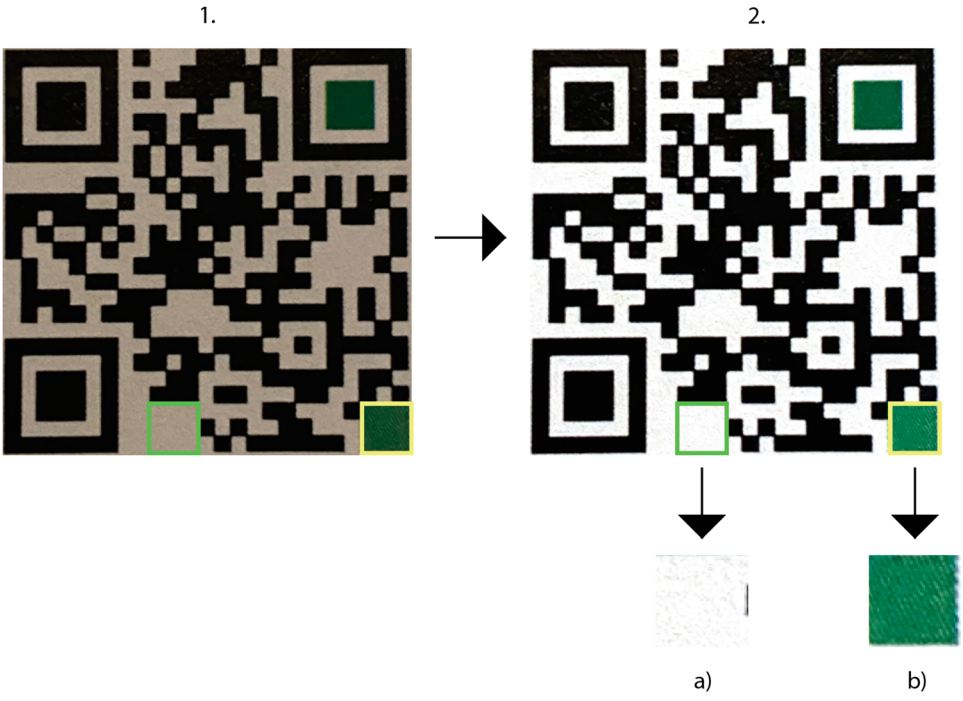


**Figure 1.** The process of color correction and area extraction.

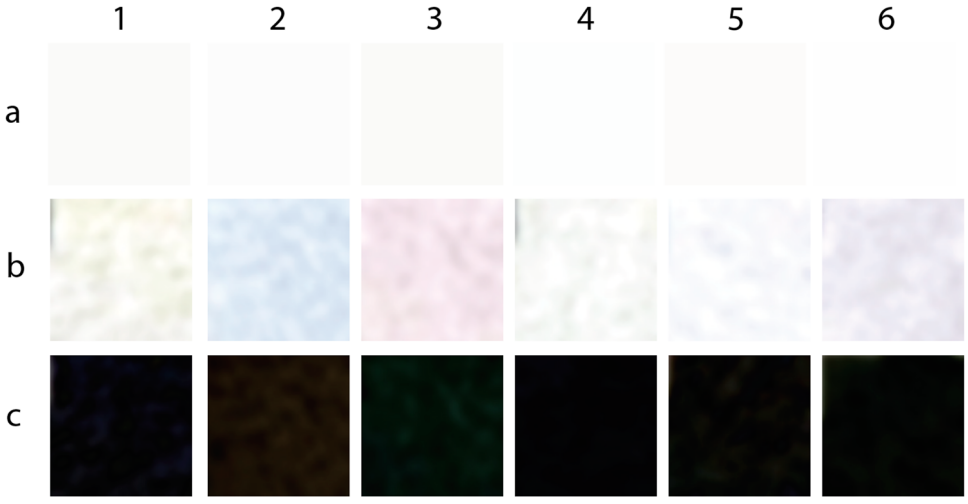


**Figure 2.** Samples of images used (best viewed online).

**Table 1.** Images per each class.

| Color | Images | Colour | Images |
|---|---|---|---|
| 10% K | 769 | 5% K | 542 |
| 10% M | 770 | 5% M | 638 |
| 10% Y | 770 | 5% Y | 650 |
| 10% CY | 818 | 5% CY | 506 |
| 10% C | 830 | 5% C | 626 |
| 0% K | 926 | | |

### 3.2. Convolutional neural networks

A Convolutional Neural Network (CNN) is a specialized type of artificial neural network specifically designed for processing and analyzing grid-structured data, such as images. Its architecture is optimized to reduce the number of parameters and computational overhead of the neural network, making it particularly well suited for the efficient handling of high-dimensional datasets [30].

The convolutional layer is the fundamental building block of a CNN (Figure 4). It applies convolutional filters (kernels) to localized regions of the input data, generating feature maps that capture essential patterns such as edges, color information, textures, and shapes. Each filter is specialized to detect a specific type of feature, and during training, the network optimizes these filters to best suit the task at hand [30].

The pooling layers in CNNs are used to down-sample feature maps, reducing their spatial dimensions and the number of parameters in the network. This reduction decreases the computational complexity and helps prevent overfitting, thereby enhancing the network's ability to generalize to new data. Common pooling methods include max pooling, which selects the maximum value within a specified window, and average pooling, which computes the average value, both contributing to feature extraction while simplifying the model [31, 32].

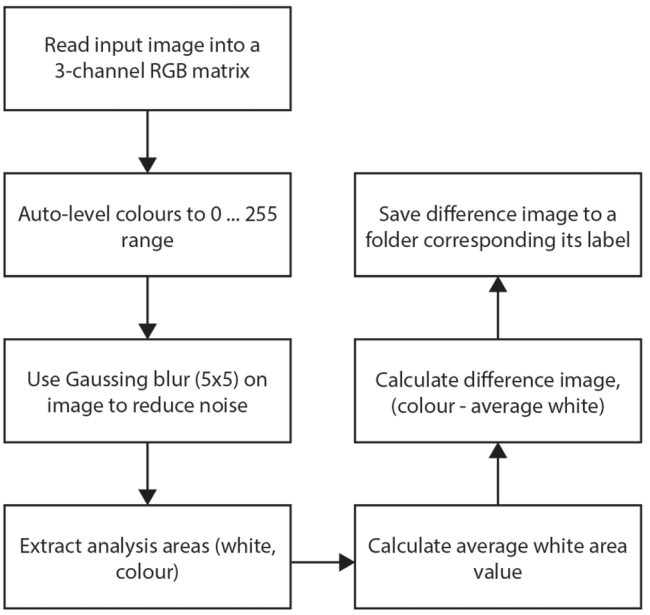


**Figure 3.** The process of color correction and area extraction.

A key feature of CNNs is the use of non-linear activation functions, such as the Rectified Linear Unit (ReLU). ReLU introduces non-linearity into the model, enabling it to capture complex patterns and relationships within grid-structured data. Furthermore, ReLU helps accelerate training convergence by alleviating the vanishing gradient problem, which is commonly encountered with activation functions such as sigmoid or tanh [30, 33].

In the final stages of a CNN, fully connected (dense) layers are used to consolidate the high-level features extracted by the convolutional layers. These layers are responsible for performing tasks such as classification or regression, using the learnt representations to generate the final output [30].

### 3.3. ResNet architecture

ResNet, short for Residual Network, was introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun from Microsoft Research in their paper titled 'Deep Residual Learning for Image Recognition', which was presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) in 2016 [1].

The ResNet architecture was the first convolutional neural network architecture to use residual learning. Residual learning addresses the vanishing gradient problem, which usually occurs when neural networks become deep. The vanishing gradient issues are closely tied to backpropagation (backward propagation of errors). In neural networks, backpropagation is a crucial element of the training process. In the backpropagation, the network learns to adjust its weights and biases to minimize the difference between its predicted output and the actual target output [34].

In backpropagation, the input data is moved through the network to produce predictions (forward pass). In this process, the difference between the predictions and the actual output (loss) is computed. During the backward pass, gradients of the loss concerning network parameters are calculated using the chain rule. These gradients are then used to guide the adjustment of weights and biases, the purpose is to minimize the loss through iterative updates. As the process is repeated for multiple epochs, it allows the neural network to learn and improve its predictive capabilities [35].

The vanishing gradient problem occurs in deep neural networks during backpropagation, as gradients are exponentially diminished in the process of backward passing through layers. In particular, the vanishing gradient problems occur in networks with many layers. Another context where vanishing gradient easily occurs are activation functions with saturating gradients, such as sigmoid or hyperbolic tangent functions. In the vanishing gradient problem gradients are approaching zero when moving closer to early layers, so these layers receive only minimal updates. As this happens, the learning process decreases, which eventually leads to slow convergence and poor performance in the training of deep networks [36, 37].

The vanishing gradient problem can easily occur in low-level features, where the difference on pixel level is small. DenseNet architecture or its modifications have been successfully used in the past to overcome problems with the low-level features [38–40].

The innovation presented in ResNet was skip connections (Figure 5), also known as shortcut connections, which skip one or more layers by adding the input of a layer to its output. This allows the network to learn residual functions instead of directly learning the underlying mapping functions. Instead of purely learning a mapping from input to output, each layer of a residual network is tasked with learning the residual function, the difference between the desired output and the input to that layer. With this approach ResNet enables the training of much deeper networks (hundreds of layers) while maintaining or improving performance [1].

Mathematically, this can be represented as follows:

$$Output = Input + Residual \tag{3}$$

When residuals are used, the network can focus on learning small incremental adjustments to the input rather than learning to generate the entire output from scratch. Residual connections enable the gradient to flow more easily during training, mitigating the problem of vanishing gradient. In practice (Figure 5), a residual block typically consists of two or more convolutional layers followed by a skip connection that adds the input to the output of the convolutional layers.

Recently ResNet has been used successfully, for example, in solving different problems. For example, Hossain et al. proposed a weighted ensemble deep transfer learning framework with ResNet152 to identify Alzheimer disease from MRI images. Hassan et al. used ResNet-50 to classify images in the Medical MNIST dataset [41]. Senapati et al. also used ResNet-50 to classify food images, Wu et al. for chicken gender identification and Lin & Wu for diabetic retinopathy detection. Madhukar et al. incorporated ResNet in cancer image segmentation [42–46]. ResNet has also been used in many other recent researches. Usually, it achieves high classification accuracy in cases where small differences play an important role.

The development of the ResNet architecture has also led to different variations, the most popular way being to change the amount of layers. For example, ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152 have been used in previous research. One of the most popular architecture of these is ResNet-50.

But also some variations have been researched, Wide ResNet increases the width of ResNet by increasing the number of channels in each convolutional layer. This can improve performance, but requires more computational resources [47].

ResNeXt introduced a new block structure that increased model capacity by aggregating multiple paths of information flow within each block [48].ResNetv2 introduced improvements to the original ResNet architecture, such as using preactivation residual units and employing a bottleneck structure in all layers [2].

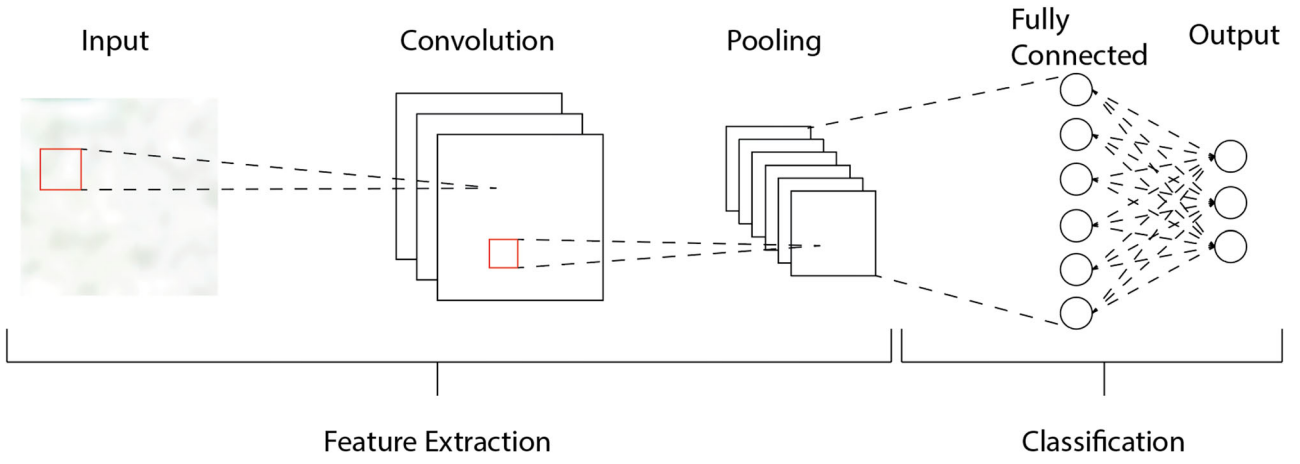


**Figure 4.** Simplified illustration of convolutional neural network.

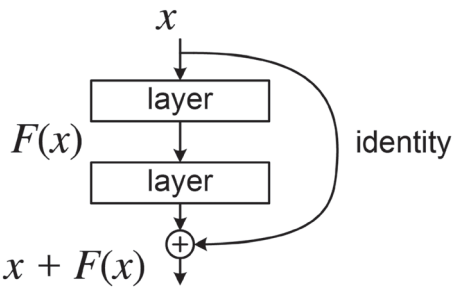


**Figure 5.** Skip connection [1].

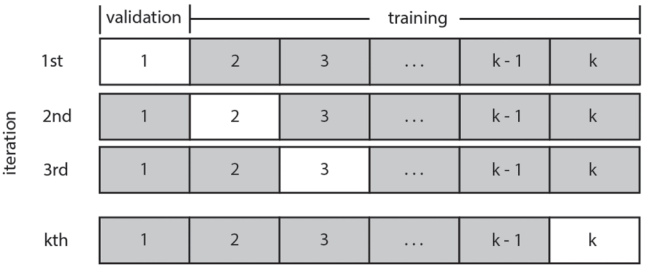


**Figure 6.** Illustration of *K*-fold cross-validation.

### 3.4. Cross-validation

In this research, the dataset used is quite small, with only 7855 images. For this reason, *K*-Fold cross-validation is used. In the *K*-Fold Cross-Validation, the training dataset is divided into *k* subsets of approximately equal size. These subsets are often called 'folds', so cross-validation is *K*-fold. With the used dataset size, the size of each subset is around 1560 images, when five folds are used. The advantage of using *K*-fold Cross-Validation is that model will be evaluated multiple times during the process. The final model when the *K*-fold cross-value process is used is generally less biased than if only one training / validation dataset is used [49, 50].

The process of the *K*-fold Cross-Validation (Figure 6) is following, before starting the process images are placed into corresponding folders, where folder represent image labels.

(1) Dataset is shuffled randomly
(2) Number of folds (*k*) is chosen and dataset is split into *k* groups
(3) For the each group:Group is taken either as hold out or test data set Rest of the groups are used as training data set Model is fitted with training data set and evaluated on the test set Evaluation score is kept and model discarded
(4) After each group is processed, model is summarized by using the sample of model evaluation scores [51]

During the process, it is important that the images remain in the same group, just because the whole group changes. During the process, this means that each image is used to train the model $k-1$ times

The advantage of *K*-Fold cross-validation is that by training the model *k* times using different combinations of training and validation sets, it is possible to obtain more reliable performance metrics than with a single train-test split. Also, when optimizing the hyperparameters, *K*-Fold cross-validation helps to choose the best hyperparameter values, without making model overfitted [52].

However, choosing the value *k* is important, typically *k* values like 2,5,10 are used [49]. The larger *k* uses more computational resources and could lead to overfitting, so the value of *k* should be kept as low as possible. In their research Yadav and Shukla have proposed that for small dataset *k* value 5–6 would provide the best results, and Wong et al. showed that cross-validation should be repeated twice for such dataset [49, 51].

### 3.5. Gradient centralization

Gradient Centralisation (GC), a method proposed by Yong et al., can be used instead of batch normalization (BN) and weight standardization (WS). As BN and WS make use of activations or weights, GC uses gradients directly and centralizes gradient vectors to have zero mean. Typically, using gradient centralization can lead to better generalization performance [53].

Batch normalization, being the most used of these, for internal normalization has the disadvantage of being an independent layer which processes data even after training. BN is usually also applied to a relatively large batch [54]. GC works directly on top of the gradient and centralizes the gradient vector so that it have zero means. The calculation of GC is done by getting the mean of each column of the gradient matrix/tensor, and then the center of each column is transferred to the zero means [55].

In the research conducted by Agarwal et al. GC was able to achieve higher accuracy of Denset models, which is a close relative of ResNet, than without GC. In this research, it was also shown that Adam's optimizer was the best option to train a neural network [56].

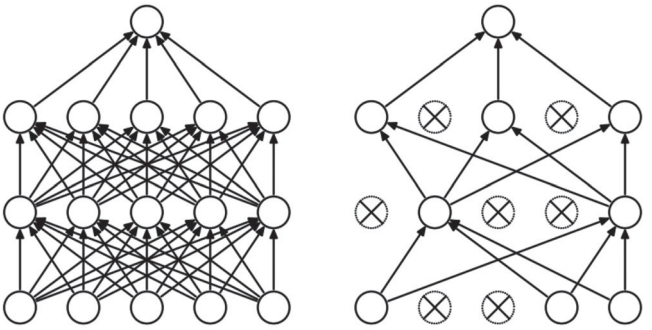


**Figure 7.** Illustration of dropout [60].

### 3.6. Dropout

Dropout can be used in convolutional neural networks to prevent the model from overfitting, also it has been shown that the use of dropout in the early phase of training can be used to avoid underfitting [57]. In the dropout, some hidden unit nodes are set stochastically to zero, which means that the nodes are removed and all forward and backward connections are also removed (Figure 7). This makes more gradient information flow through non-linear activation functions [58].

Dropout works show that during the training process, the nodes are dropped with a dropout probability of $p$ [59, 60]. The probability $p$ can vary, depending on the location of the dropout in the architecture. The probability is different if the dropout layer is placed close to the input or closer to the output. Especially dropout has been proven to work in connection with fully connected layers [61]. During the training process, probability can be seen as one hyper-parameter which can be adjusted to find the optimal model. The dropout rate is usually smaller ($p < 0.2$) in input layer and larger ($0.5 < p < 0.8$) when used internally or close to a fully connected layer [62]. Some research has also proposed a more controlled way to use dropout [63].

Dropout layers can be easily added to any architecture; this has some consequences. When dropout is added, it removes some units, which reduces the capacity of the network; this can be compensated by adjusting the number of units used by multiplying them by $1/(dropoutrate)$. As dropout introduces noise to the gradient, it has been shown that increasing the learning rate and momentum is needed; however, this depends on the optimizer used. If such a modification is made, max norm regularization might be needed [62].

When a dropout layer is added to an architecture it also makes it possible to fine-tune hyperparameters that relate. Depending on the used optimizer learning rate, weight decay and momentum parameters can be tuned, and this is usually even required to get the model to work in a optimized way. Hyperparameters are important especially if the standard stochastic descent gradient (SGD) optimizer is used instead of adaptive optimizers [62].

Regarding the development of different dropout methods, past research has considered many of those, for example [64–68]. In this research, standard version, where nodes area randomly dropped, is used.

### 3.7. Max and average pooling

The pooling layers are used in convolutional neural networks for downsampling. Currently, the most common pooling layers are max and average pooling. In addition, other pooling methods have been invented. The purpose of downsampling is to reduce the spatial resolution of feature maps to extract semantic information [69].

Pooling layers have two main objectives: (1) they reduce the number of parameters, which makes training of the networks less expensive, and (2) they help in avoiding possible overfitting of the network [70].

Pooling can either be done based on local regions, like $3 \times 3$, $5 \times 5$ or $7 \times 7$ areas. Another option is to use global pooling, where each feature maps across all spatial locations, resulting in a global representation of the features that summarizes the entire input volume [70].

When the values are calculated in the pooling layer (Figure 8), the maximum or average pooling is used. Maximum pooling uses the greatest value within a given region ($k \times k$) and pools it into the corresponding region in the downsampled feature map. Average pooling calculates the average value within a given region ($k \times k$) and uses this value in the downsampled feature map [70].

Using maximum or average pooling depends on the problem. In most cases, convolutional neural networks (CNNs) are looking to recognize significant values in images. This would make maximum pooling a preferable option, but average pooling is better at preserving localization [71].

## 4. Experiments and results

The experiments were run in a Python environment, with a laptop with an Apple M2 processor, which had 8 cores and a total RAM of 16 Gt. The environment used Python version 3.9.6 with Tensorflow and Keras 2.15.0. The architectures used were built manually on the basis of the ResNet architecture documentation and examples available.

The training process of the models was done with Adam optimizer, with default settings as the purpose of this research was not to focus optimizing individual models via hyper-parameters. For each of the training processes, 30 epochs were used.

### 4.1. Evaluation metrics

The purpose of the experiments was to find out which approach achieves the best accuracy. Accuracy is an effective performance metric when applied to datasets with balanced class distributions. A balanced dataset ensures that each class is equally represented, minimizing the risk that performance metrics are disproportionately influenced by the prevalence of a particular class. In these scenarios, accuracy offers a clear and straightforward measure of the proportion of correctly classified instances in the total number of instances, capturing the overall predictive capacity of the model. Accuracy serves as an appropriate metric when the application context assigns equal importance to all classes in the dataset. In classification problems where the costs or implications of misclassifications are uniform across all classes, accuracy provides a meaningful and holistic evaluation of the model's performance.

Accuracy can be calculated with the following equation:

$$Accuracy = \frac{Number\ Of\ Correct\ Predictions}{Total\ Number\ Of\ Predictions} \tag{4}$$

In the experiments, the accuracy was calculated for the entire dataset.

In addition, a confusion matrix was created for a detailed breakdown of correct and incorrect predictions for each class in a classification problem. A confusion matrix is an effective tool for evaluating the performance of classification models. Confusion matrix provides a detailed comparison of predicted versus actual outcomes, enabling the assessment of overall accuracy and the identification of specific error patterns. This analysis can be used to improve the

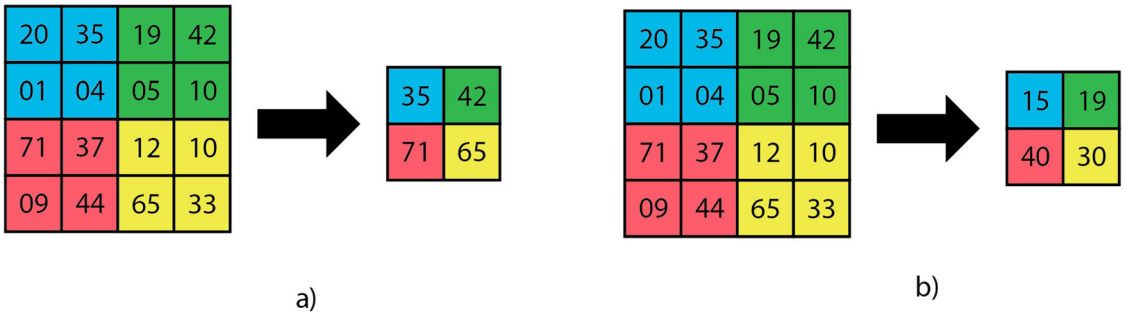


**Figure 8.** (a) Max pooling and (b) average pooling.

model architecture, optimize preprocessing methods, and improve data acquisition strategies [72, 73]

A confusion matrix is typically presented as a structured square table with rows and columns representing different classes in the classification task. For a binary classification problem (labelled as Positive and Negative), the matrix is a 2 × 2 table:

| | Predicted Positive | Predicted Negative |
|---|---|---|
| Actual Positive | True Positive(TP) | False Negative (FN) |
| Actual Negative | False Positive (FP) | True Negative (TN) |

True Positive (TP) refers to instances correctly identified as positive, while True Negative (TN) corresponds to instances accurately classified as negative. False Positive (FP) represents instances incorrectly predicted as positive, and False Negative (FN) denotes instances incorrectly predicted as negative.

In multiclass classification problems, the confusion matrix extends the binary version into a square matrix, with rows and columns representing the various classes. This structure provides a detailed view of the performance of the model, showcasing accurate classifications and misclassifications in all categories.

For a classification problem with $n$ classes, the confusion matrix will be an $n \times n$ matrix. Each element in the matrix at position $(i, j)$ represents the number of instances where the true class is $i$, and the predicted class is $j$. The diagonal elements represent the number of correct predictions for each class, and the off-diagonal elements represent misclassifications.

In Figure 9, 13 images of *label 1* were classified as *label 1*, and no images were classified as other labels. Then, in *label 2*, some images have been labeled as *label 3*. Furthermore, all images of *label 3* are correctly labeled. Therefore, the model may require some improvement.

A confusion matrix offers a more nuanced understanding of a classification model's performance compared to scalar metrics such as accuracy. The confusion matrix identifies the specific classes that the model tends to confuse and highlights strengths and weaknesses in its predictions. This detailed analysis provides critical information for the refinement of model design, the optimization of training processes, and the addressing of specific areas for improvement.

### 4.2. Standard ResNet architectures

As the purpose of the experiment was to find the best model for the use case, the fine-tuning of the hyperparameters and choosing the best optimizer were left for another research. All models were compiled with categorical cross-entropy loss and Adam optimizer (Table 2).

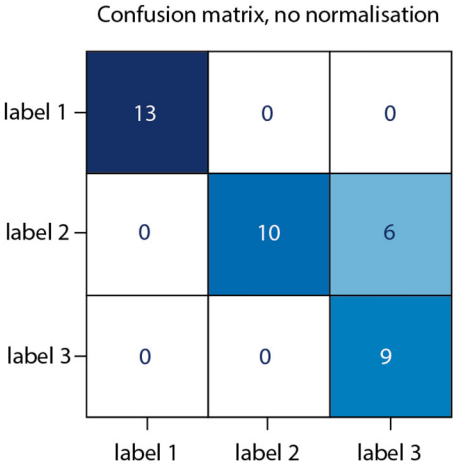


**Figure 9.** Example of confusion matrix for multi-classification.

**Table 2.** Results of first cycle.

| Architecture | Centralised gradient used | avg. train time/epoch (s.) | Accuracy |
|---|---|---|---|
| ResNet-18 | Y | 323 s. | 0.934 |
| ResNet-18 | N | 316 s. | 0.952 |
| ResNet-34 | Y | 236 s. | 0.969 |
| ResNet-34 | N | 233 s. | 0.966 |
| ResNet-50 | Y | 785 s. | 0.958 |
| ResNet-50 | N | 781 s. | 0.901 |
| ResNet-101 | Y | 1414 s. | 0.941 |
| ResNet-101 | N | 1350 s. | 0.948 |
| ResNet-153 | Y | 2079 s. | 0.682 |
| ResNet-153 | N | 1094 s. | 0.935 |

From the experiments (Table 3) carried out with the standard implementations of ResNet, it can be seen that all implementations perform well. Using gradient centralization improves results in ResNet-50 and ResNet-34. But when architecture is deep, centralization of gradients decreases accuracy. When using the ResNet-153 architecture, the centralization of the gradients significantly reduces the accuracy than without it. This might be an indication that when the differences are small in the data, the vanishing gradient problem becomes a problem with gradient centralization.

Most of the ResNet architectures were prone to overfitting in the given problem, especially architectures deeper than ResNet-34. Overfitting of the model signals that training goes well, but the model fails to generalize to the validation set. In practice, this means that the model learns the training data too well, capturing noise and specifics of the training set that do not generalize to new, unseen data [74].

**Table 3.** *K*-fold accuracy.

| Fold | Architecture (a) | ResNet-34 | Architecture (b) |
|---|---|---|---|
| 0 | 97.66% | 95.31% | 97.85% |
| 1 | 99,51% | 98.93% | 99.22% |
| 2 | 96.58% | 97.66% | 97.17% |
| 3 | 96.29% | 98.73% | 98.54% |
| 4 | 100.00% | 96.58% | 98.63% |
| Final | 98.00% | 97.44% | 98.28% |

The small dataset can also be an issue, as the dataset contains only 7855 images. For extending the dataset, more images could be collected or different augmentation options could be used [75].

Of all architectures tested, ResNet-34 with gradient centralization had the best accuracy, as the model reached 96.9% accuracy. And it was selected for fine-tuning the architecture.

In Figure 10, it can be seen that the training accuracy of ResNet-34 improves throughout the epochs. In addition, training loss decreases during the training process. After 10 epochs, validation accuracy starts to behave unexpectedly and drops, while training accuracy seems to settle. At the end of the training process, the training loss seems to increase. Some of the reasons behind this could be overfitting, a small dataset, inappropriate learning, or simply that the model has already reached its optimal performance. To overcome these and build better models, ResNet-34 was modified.

If the model reaches its peak performance, it might be wise to implement early stopping for the training process; this prevents overfitting and saves computation time when further training does not yield significant improvements [76].

### 4.3. Proposed architecture

The base model of ResNet-34 was able to reach the accuracy of 96.90%, this is already very high, but with some modifications it might be possible to reach even higher accuracy. Some approaches were first experimented individually and finally, all of them were combined.

One of the approaches used in past research to improve CNN is adding a dropout layer to fully connected layers. With the usage of dropout, it is possible to reduce overfitting and regulate neural networks [60]. Proper usage of dropout layers can increase the accuracy of neural networks [77]. The traditional approach to using dropout is to add dropout after the convolutional layer; however, there are different variations of using dropout [78]. In this research, two variations of the use of dropout in the ResNet-34 architecture were used. In the first dropout, it was added after the first convolutional layer. This convolutional layer pools feature maps down. Then, the dropout was added after each residual group. The performance of these architectures was 98.90% and 94.34% accordingly. This shows that dropping out at the correct location in the architecture can make the model perform more accurately.

Adding batch normalization can also make the network more resistant towards degradation of the between-class distance to the within-class distance ratio. With batch normalization, it is possible to perceive the between-class angle [79]. When batch normalization was added at the end of the residual block accuracy of the model dropped 1.0% from standard ResNet-34 implementation.

Another option to improve the accuracy of the CNN is to use different grouping layers. Pooling layers play a crucial role in reducing the spatial dimensions of the feature maps, controlling the overfitting, providing translation invariance, and facilitating feature learning and abstraction in convolutional neural networks [80]. Local pooling, which is used to pool data from smaller regions, or global pooling, which works on the feature map, can be used to improve the accuracy and sensitivity of feature translation [70]. With the usage of different pooling methods, it might be possible to increase the accuracy of CNN architecture, as, for example, in the research by Momeny et al. [81].

In the residual block of ResNet after each convolutional operation, a maximum pooling layer was added, which led to a accuracy of 95.10%. When max-pooling was added only after both convolutional operations were performed, residual block accuracy reached as high as 99.70%. As a last modification to the standard ResNet-34 architecture just before the fully connected layer average pooling was changed to global max pooling instead of average pooling 99.60%

When finally all the best options from the above experiments were combined, using dropout after input layers, the global maximum pooling at the end of the residual block and changing the average pooling to the global maximum pooling accuracy was dropped to 40.52%. This shows that even if optimisations work independently they don't work together. In addition, some combinations were also tested to see if they can reach a higher accuracy than individual modifications. Using global maximum pooling instead of average pooling with dropout performed with 99.20% accuracy. Using global maximum pooling instead of average pooling with a maximum pooling layer at the end of the residual block reached 99.50% accuracy.

For the initial experimentation, without cross-validation, training and validation accuracy and loss, together with confusion matrixes, are shown in Figure 11. For both of the architectures it can be seen that models are learning quite well throughout the epochs, and if 30 epochs are used, early stopping does not stop the training process. Architectures reach maximum accuracy at the end of the process (architecture a 99.70% and b 99.60%). In both cases, the training loss seems to stabilize at the end of the training process.

Some mistakes that version a) makes are that low-level green (10% CY) and yellow (10% Y) images are confused with very low yellow-color (5% Y) images. With this approach, it was possible to reach an accuracy of 99.7%. Version b) of the architecture almost reaches the same accuracy (99.6%). In this architecture, there was more confusion between classes. Different green colors are confused (5% CY and 10% CY), as well as different yellow classes. Also, very light yellow is sometimes confused with paper white.

Based on the experiments, two best options to improve the ResNet-34 architecture were:

- (a) changing the average pooling to the max pooling before fully connected layer.
- (b) using the maximum pooling at the end of the residual block. These architectures are presented in Figure 12.

### 4.4. K-fold cross-validation

The accuracy of both proposed architectures was finally verified with *K*-Fold cross-validation using five folds. For each fold 30 epochs of training were run. The same process was performed for the standard ResNet-34 architecture in order to obtain the accuracy for comparison. The following table summarizes these results. The final accuracy in *K*-Fold cross-validation is calculated by averaging the accuracy scores obtained from each fold.

The results show that architecture (b), where no other changes were made than changing the average pooling to the maximum pooling, is the most accurate of all versions tested. In Figure 13 the training and validation loss through epochs for each fold can be seen. The figures show that model reaches its peak accuracy typically between 20 and 30 epochs, after that model starts to show signs over overfitting.


**Figure 10.** Results of the train and validation process of ResNet-34.

Finally, Figure 14 show the combined confusion matrix of *K*-Fold cross-validation. In this figure, values are normalized so that it can easily be observed how well the model perform. The confusion matrix for *K*-Fold cross-validation shows similar results to those without *K*-Fold cross-validation. The most challenging class to identify is 5% Y, this is confused with white paper in 8% of cases, also sometimes 10% and 5% Y are confused. Other classes are rarely confused.

## 5. Discussion

This article presented two optimized versions of ResNet, with the modified version, where a maximum pooling layer was added after convolutional operations, the residual block accuracy was 98.00% in the cross-validation of the *K*-Fold. The second modification was to change the last average pooling layer to the maximum pooling layer, this version of the ResNet reached 98.28% accuracy being the most accurate version. The results show that ResNet can be used to recognize subtle color differences. The mentioned modifications make it even better for the use case, as both versions outperformed the standard ResNet-34 architecture.

With the proposed approach, it is possible to implement a more accurate color-based classification system that can help, for example, in civil engineering [82–84], food production [85–87] and healthcare [88, 89]. In these and some other use cases, color and color differences play an important role.

The findings of this research support previous research like that by Sukhetha et al. and Goudha et al., showing that the ResNet architecture works well when color is a criterion in the classification task [90, 91]. Singh et al. [92] have also shown in their research that convolutional networks are highly color dependent. This dependency can be seen in the presented research; CNNs can learn to classify color even with shallow networks. Modifications to the standard ResNet architecture are a good way to make ResNet suitable for various color recognition tasks; this is in line with previous research such as Mathew et al. [93], Zhang et al. [94] and Reddy et al. [95].

One limitation of this research is that it uses a relatively small dataset of images which are printed with a specific color printer. This might have an impact on the results presented; however, the presented approach can be adjusted to different image sources with more training and possibly by adjusting preprocessing methods for images.

The proposed solution was run with a standard laptop environment; more research would be needed about how to make the system run on mobile devices, if color recognition is wanted to be done in real-time, for example, in the consumer context. The use of extended datasets and different preprocessing algorithms might also make the proposed approach more generalizable. Deploying the system in a server-client infrastructure involves a client-side application, such as a Web or mobile app, for user interaction and image uploads, paired with a server-side back-end hosting the trained ResNet-34 model for pre-processing and image classification. The trained model can be deployed for web with frameworks such as TensorFlow together with the help of Flask. In such an application, continuous training of

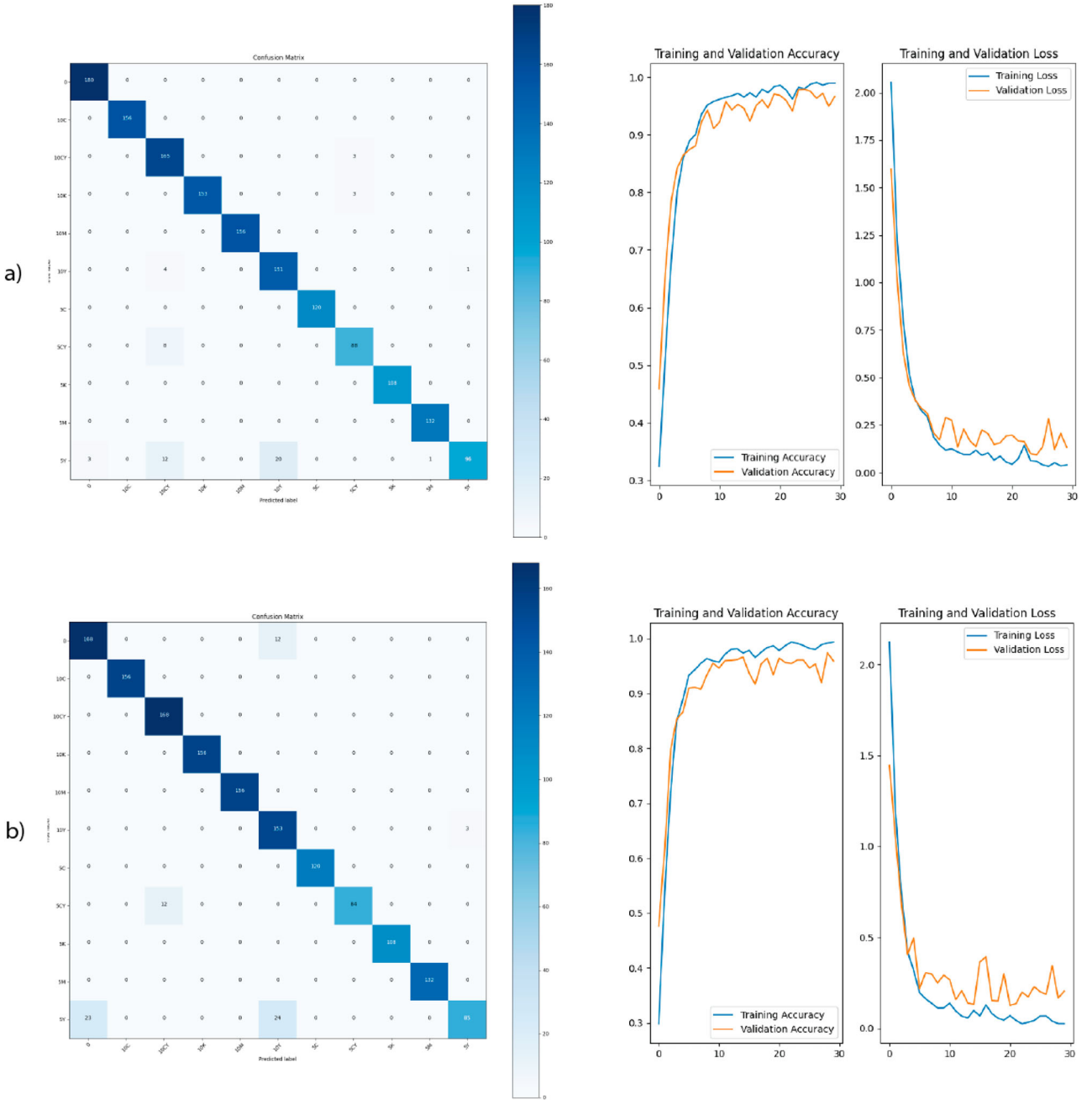


**Figure 11.** Results of the process.

the model might also be a feasible feature and might lead to a more accurate solution.

Future research could also focus on tailoring the model for specific industrial applications, such as automating quality assurance in manufacturing or improving diagnostic capabilities in telemedicine. This involves optimizing the system to meet operational constraints, such as processing speed, scalability, and integration with existing workflows. In preprocessing, exploring alternative preprocessing techniques, such as adaptive histogram equalization or domain-specific adjustments, could further enhance the robustness and reliability of the model under varying lighting and environmental conditions, making it more adaptable to real-world scenarios. In addition, creating diverse datasets with variations in imaging conditions, color ranges, and patterns would help generalize the system to a wider variety of use cases and ensure its effectiveness in different domains.

## 6. Conclusions

The research highlights the effectiveness of different ResNet architectures for subtle color classification and demonstrates how targeted modifications can enhance model performance. By systematically evaluating different ResNet variants, the research identified ResNet-34 as the most suitable baseline model, with gradient centralization further enhancing its accuracy. Two custom versions of ResNet-34 were proposed, achieving a classification accuracy of up to 99.28% through the use of global max pooling instead of global average pooling. The results of the experiment were verified with a 5-fold $K$-Fold cross-validation. These results underscore the flexibility of ResNet architectures and the importance of fine-tuning their components for specific applications. The proposed architectures demonstrate potential for real-world applications in fields like civil engineering, food production, and healthcare, where precise color differentiation plays

| Layer name | output size | custom Resnet - version a |
|---|---|---|
| conv1 | 112 x 112 | 7x7, 64 stride 2 |
| conv2 | 56 x 56 | 3x3 max pool, stride 2 |
| | | [3 x 3, 64<br>3 x 3, 64<br>3x3 max pool, stride 2] x 3 |
| | | [3 x 3, 64<br>3 x 3, 64] x 1 |
| conv3 | 28 x 28 | [3 x 3, 128<br>3 x 3, 128<br>3x3 max pool, stride 2] x 3 |
| | | [3 x 3, 128<br>3 x 3, 128] x 1 |
| conv4 | 14 x 14 | [3 x 3, 256<br>3 x 3, 256<br>3x3 max pool, stride 2] x 5 |
| | | [3 x 3, 256<br>3 x 3, 256] x 1 |
| conv5 | 7x 7 | [3 x 3, 256<br>3 x 3, 256<br>3x3 max pool, stride 2] x 3 |
| | 1 x 1 | average pool, fc, softmax |

| Layer name | output size | custom Resnet - version b |
|---|---|---|
| conv1 | 112 x 112 | 7x7, 64 stride 2 |
| conv2 | 56 x 56 | 3x3 max pool, stride 2 |
| | | [3 x 3, 64<br>3 x 3, 64] x 3 |
| | | [3 x 3, 64<br>3 x 3, 64] x 1 |
| conv3 | 28 x 28 | [3 x 3, 128<br>3 x 3, 128] x 3 |
| | | [3 x 3, 128<br>3 x 3, 128] x 1 |
| conv4 | 14 x 14 | [3 x 3, 256<br>3 x 3, 256] x 5 |
| | | [3 x 3, 256<br>3 x 3, 256] x 1 |
| conv5 | 7x 7 | [3 x 3, 256<br>3 x 3, 256<br>3x3 max pool, stride 2] x 3 |
| | 1 x 1 | maximum pool, fc, softmax |

**Figure 12.** Final architectures used.

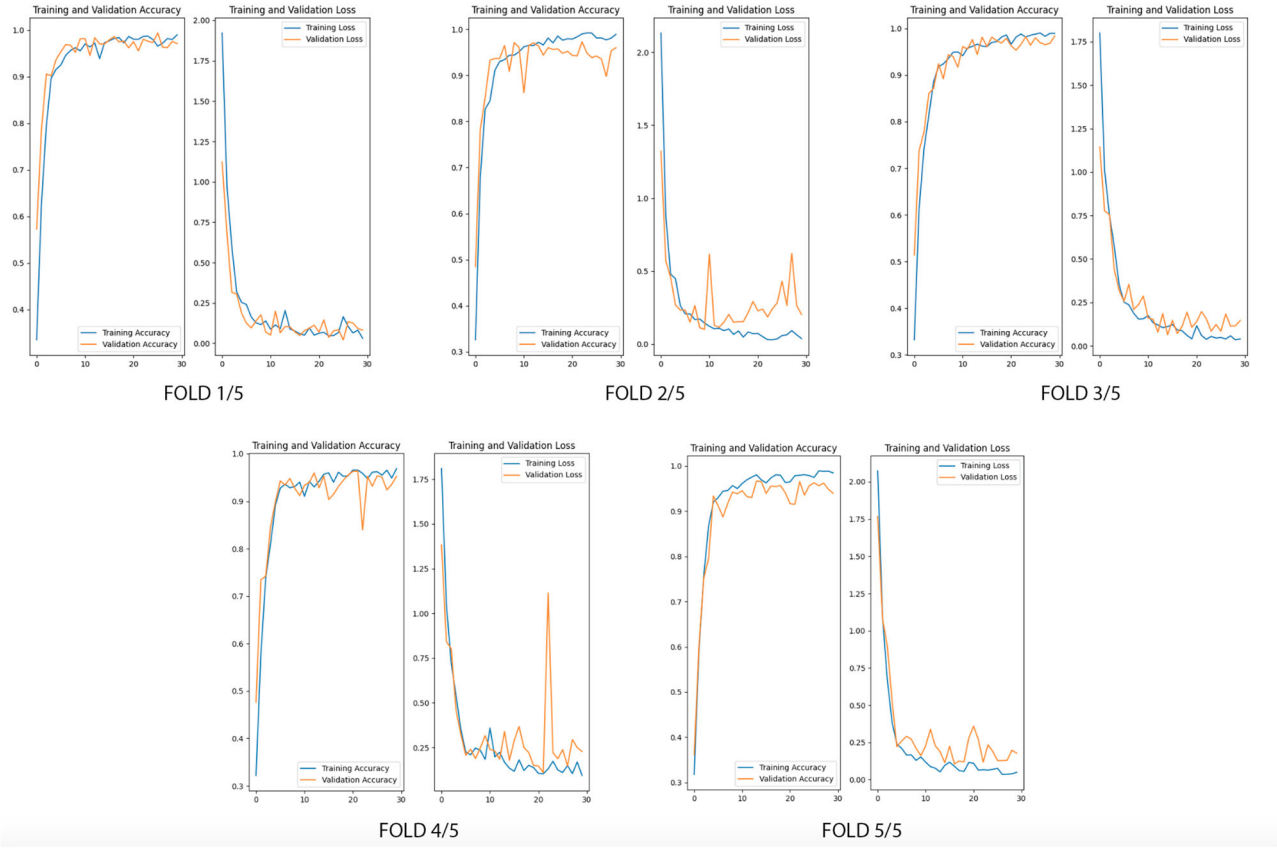


**Figure 13.** Training process of each fold.

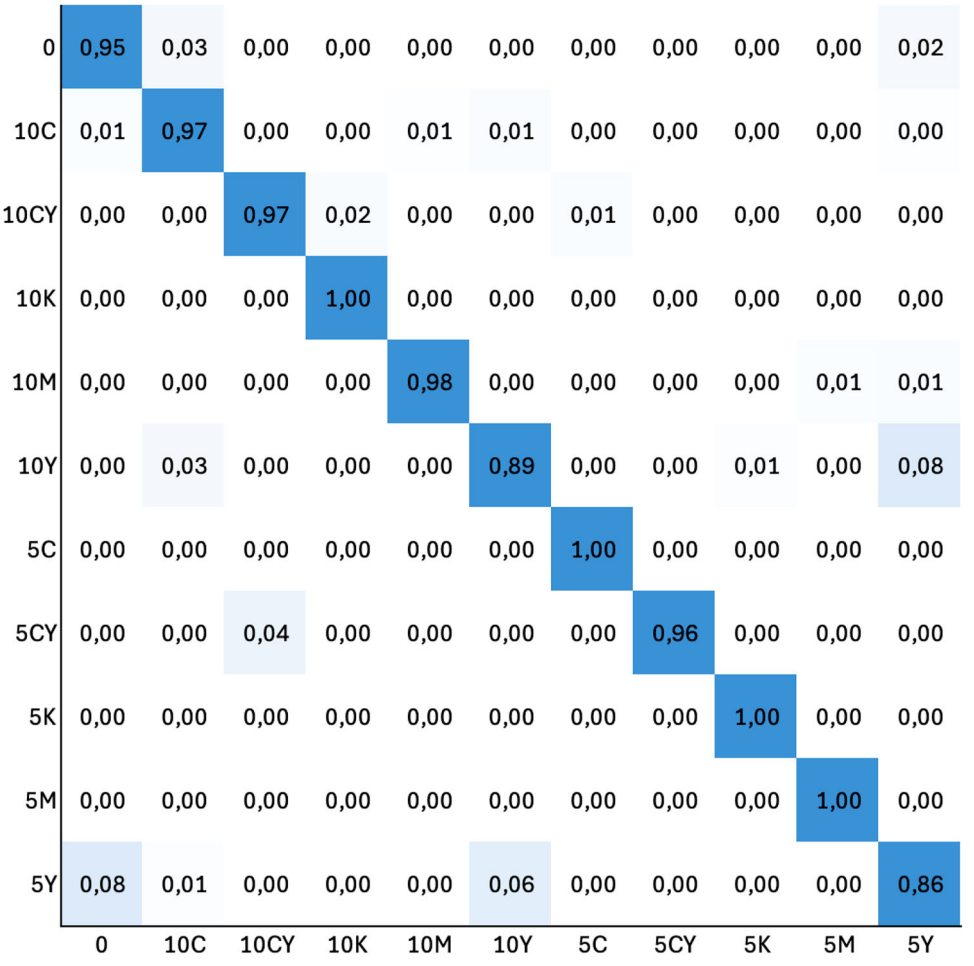


**Figure 14.** Combined confusion matrix of *K*-fold cross-validation.

a crucial role. Some restrictions of the experiment were small dataset, that was augmented and usage of one printer type to print QR-code that worked as color information carries. Future work around the topic could explore broader datasets, alternative preprocessing methods, and real-time implementation on mobile platforms to extend the applicability of the proposed approaches.

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

This work was supported by the Finnish Cultural Foundation's Central Ostrobothnia Regional Fund (Suomen Kulttuurirahasto) [grant number 2521 1242].

## Data availability statement

One of the datasets used in this manuscript is available as Zenodo repository: https://doi.org/10.5281/zenodo.11079897.

## Notes on contributor

***Jari Isohanni***, MSc (Computer Science), is currently working with his doctoral studies at the University of Vaasa (Digital Economy). His dissertation compares different approaches in recognition of subtle color differences in printed sources. Jari has been working in the software industry since 2004, currently acting as Director (Education) at Centria University of Applied Sciences.

## ORCID

*Jari Isohanni* http://orcid.org/0000-0002-7154-2515

## References

[1] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA; 2016. p. 770–778.
[2] He K, Zhang X, Ren S, et al. Identity mappings in deep residual networks. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. Springer; 2016. p. 630–645.
[3] He K, Zhang X, Ren S, et al. Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA; 2015. p. 1026–1034.
[4] Yang S, Liu S, Yang C, et al. Re-rank coarse classification with local region enhanced features for fine-grained image recognition. Preprint; 2021. arXiv:210209875.
[5] Ailing Q, Ning T. Fine-grained vehicle recognition method based on improved ResNet. In: 2nd International Conference on Information Technology and Computer Application (ITCA) IEEE; 2020. p. 588–592.
[6] Li S, Jiao J, Han Y, et al. Demystifying ResNet. Preprint; 2016. arXiv:16110 1186.
[7] Targ S, Almeida D, Lyman K. Resnet in ResNet: generalizing residual architectures. Preprint; 2016. arXiv:160308029.
[8] Zagoruyko S, Komodakis N. Wide residual networks. Preprint; 2016. arXiv:160507146.
[9] Li B, He Y. An improved ResNet based on the adjustable shortcut connections. IEEE Access. 2018;6:18967–18974. doi: 10.1109/ACCESS.2018.2814605
[10] Yuan P, Lin S, Cui C, et al. HS-ResNet: hierarchical-split block on convolutional neural network. Preprint; 2020. arXiv:201007621.

[11] Hayou S, Clerico E, He B, et al. Stable ResNetA Virtual Conference International Conference on Artificial Intelligence and Statistics; 2021. p. 1324–1332.
[12] Bharati S, Podder P, Mondal M, et al. Co-ResNet: optimized ResNet model for COVID-19 diagnosis from X-ray images. Int J Hybrid Intell Syst. 2021;17(1–2):71–85.
[13] Wen L, Li X, Gao L. A transfer convolutional neural network for fault diagnosis based on ResNet-50. Neural Comput Appl. 2020;32:6111–6124. doi: 10.1007/s00521-019-04097-w
[14] Sarwinda D, Paradisa RH, Bustamam A, et al. Deep learning in image classification using residual network (ResNet) variants for detection of colorectal cancer. Procedia Comput Sci. 2021;179:423–431. doi: 10.1016/j.procs.2021.01.025
[15] Li B, Lima D. Facial expression recognition via ResNet-50. Int J Cogn Comput Eng. 2021;2:57–64.
[16] Yu X, Kang C, Guttery DS, et al. ResNet-SCDA-50 for breast abnormality classification. IEEE/ACM Trans Comput Biol Bioinform. 2020;18(1):94–102. doi: 10.1109/TCBB.8857
[17] Gao M, Qi D, Mu H, et al. A transfer residual neural network based on ResNet-34 for detection of wood knot defects. Forests. 2021;12(2):212. doi: 10.3390/f12020212
[18] Hammad M, Pławiak P, Wang K, et al. Resnet-attention model for human authentication using ECG signals. Expert Syst. 2021;38(6):e12547. doi: 10.1111/exsy.v38.6
[19] Han C, Shi L. ML–ResNet: a novel network to detect and locate myocardial infarction using 12 leads ECG. Comput Methods Programs Biomed. 2020;185:105138. doi: 10.1016/j.cmpb.2019.105138
[20] Isohanni J. Use of functional ink in a smart tag for fast-moving consumer goods industry. J Packag Technol Res. 2022;6(3):187–198. doi: 10.1007/s41783-022-00137-4
[21] Almoosawi NM, Khudeyer RS. ResNet-34/DR: a residual convolutional neural network for the diagnosis of diabetic retinopathy. Informatica. 2021;45(7):115–124.
[22] Hu WJ, Fan J, Du YX, et al. MDFC–ResNet: an agricultural IoT system to accurately recognize crop diseases. IEEE Access. 2020;8:115287–115298. doi: 10.1109/Access.6287639
[23] Al-Haija QA, Adebanjo A. Breast cancer diagnosis in histopathological images using ResNet-50 convolutional neural network. In: IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Vancouver, BC, Canada. IEEE; 2020. p. 1–7.
[24] Zhang X, Li H, Sun S, et al. Classification and identification of apple leaf diseases and insect pests based on improved ResNet-50 model. Horticulturae. 2023;9(9):1046. doi: 10.3390/horticulturae9091046
[25] Shaheed K, Qureshi I, Abbas F, et al. EfficientRMT-Net – an efficient ResNet-50 and vision transformers approach for classifying potato plant leaf diseases. Sensors. 2023;23(23):9516. doi: 10.3390/s23239516
[26] Li X, Rai L. Apple leaf disease identification and classification using ResNet models. In: 3rd International Conference on Electronic Information and Communication Technology (ICEICT); Shenzhen, China. IEEE; 2020. p. 738–742.
[27] Isohanni J. QR-codes with colour embed inside. Zenodo. 2024. doi:10.5281/zenodo.11079897.
[28] Basuki A, Ramadijanti N. Improving auto level method for enhancement of underwater images. In: Manado International Conference on Knowledge Creation and Intelligent Computing (KCIC); 2016; p. 120–125. doi:10.1109/KCIC.2016.7883635.
[29] Zhang G, Lin L, Wang J. Lung nodule classification in CT images using 3D DenseNet. J Phy Conf Series 2021;1827:012155.
[30] Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge, USA: MIT Press; 2016.
[31] LeCun Y, Boser B, Denker J, et al. Handwritten digit recognition with a backpropagation network. Adv Neural Inf Process Syst. 1989;2:396–404.
[32] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–2324. doi: 10.1109/5.726791
[33] Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10); 2010; p. 807–814.
[34] Goh GB, Hodas NO, Vishnu A. Deep learning for computational chemistry. J Comput Chem. 2017;38(16):1291–1307. doi: 10.1002/jcc.v38.16
[35] Zweiri YH, Whidborne JF, Seneviratne LD. A three-term backpropagation algorithm. Neurocomputing. 2003;50:305–318. doi: 10.1016/S0925-2312(02)00569-6
[36] Wang X, Qin Y, Wang Y, et al. ReLTanh: an activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis. Neurocomputing. 2019;363:88–98. doi: 10.1016/j.neucom.2019.07.017
[37] Rehmer A, Kroll A. On the vanishing and exploding gradient problem in gated recurrent units. IFAC-PapersOnLine. 2020;53(2):1243–1248. doi: 10.1016/j.ifacol.2020.12.1342
[38] Tong T, Li G, Liu X, et al. Image super-resolution using dense skip connections Venice. In: Proceedings of the IEEE International Conference on Computer Vision; 2017. p. 4799–4807.
[39] Ooi YK, Ibrahim H, Mahyuddin MN. Enhanced dense space attention network for super-resolution construction from single input image. IEEE Access. 2021;9:126837–126855. doi: 10.1109/ACCESS.2021.3111983
[40] Haider A, Arsalan M, Choi J, et al. Robust segmentation of underwater fish based on multi-level feature accumulation. Front Mar Sci. 2022;9:1010565. doi: 10.3389/fmars.2022.1010565
[41] Hassan E, Hossain MS, Saber A, et al. A quantum convolutional network and ResNet (50)-based classification architecture for the MNIST medical dataset. Biomed Signal Process Control. 2024;87:105560. doi: 10.1016/j.bspc.2023.105560
[42] Md Rabiul Hasan ABMAH, Ullah SMA. Ensemble ResDenseNet: Alzheimer's disease staging from brain MRI using deep weighted ensemble transfer learning. Int J Comput Appl. 2024;46(7):539–554. doi: 10.1080/1206212X.2024.2380648 .
[43] Senapati B, Talburt JR, Naeem AB, et al. Transfer learning based models for food detection using ResNet-50. In: Romeoville IEEE International Conference on Electro Information Technology (EIT); 2023. p. 224–229.
[44] Wu D, Ying Y, Zhou M, et al. Improved ResNet-50 deep learning algorithm for identifying chicken gender. Comput Electron Agric. 2023;205:107622. doi: 10.1016/j.compag.2023.107622
[45] Lin CL, Wu KC. Development of revised ResNet-50 for diabetic retinopathy detection. BMC Bioinform. 2023;24(1):1–18. doi: 10.1186/s12859-022-05124-9
[46] Madhukar BN, Bharathi SH, Polnaya AM. Multi-scale convolution based breast cancer image segmentation with attention mechanism in conjunction with war search optimization. Int J Comput Appl. 2023;45(5):353–366. doi: 10.1080/1206212X.2023.2212945 .
[47] Zagoruyko S, Komodakis N. Wide residual networks; 2017. Available from: https://arxiv.org/abs/1605.07146 [cs.CV].
[48] Xie S, Girshick R, Dollár P, et al. Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 1492–1500.
[49] Wong TT, Yeh PY. Reliable accuracy estimates from *k*-fold cross validation. IEEE Trans Knowl Data Eng. 2019;32(8):1586–1594. doi: 10.1109/TKDE.69
[50] James G, Witten D, Hastie T, et al. An introduction to statistical learning. Vol. 112. New York, USA: Springer; 2013.
[51] Yadav S, Shukla S. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In: IEEE 6th International Conference on Advanced Computing (IACC); 2016. p. 78–83.
[52] Lyu Z, Yu Y, Samali B, et al. Back-propagation neural network optimized by *k*-fold cross-validation for prediction of torsional strength of reinforced concrete beam. Materials. 2022;15(4):1477. doi: 10.3390/ma15041477
[53] Yong H, Huang J, Hua X, et al. Gradient centralization: a new optimization technique for deep neural networks. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16. Springer; 2020. p. 635–652.
[54] Fuhl W, Kasneci E. Weight and gradient centralization in deep neural networks. Preprint; 2020. arXiv:201000866.
[55] Lu F, Niu R, Zhang Z, et al. A generative adversarial network-based fault detection approach for photovoltaic panel. Appl Sci. 2022;12(4):1789. doi: 10.3390/app12041789
[56] Agarwal V, Lohani M, Bist AS. Comparative analysis of deep learning models for various optimizer embedded with gradient centralization. Int J Intell Syst Appl Eng. 2024;12(15s):445–454.
[57] Liu Z, Xu Z, Jin J, et al. Dropout reduces underfitting. In: International Conference on Machine Learning. PMLR; 2023. p. 22233–22248.
[58] Hahn S, Choi H. Understanding dropout as an optimization trick. Neurocomputing. 2020;398:64–70. doi: 10.1016/j.neucom.2020.02.067
[59] Wu H, Gu X. Towards dropout training for convolutional neural networks. Neural Netw. 2015;71:1–10. doi: 10.1016/j.neunet.2015.07.007
[60] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–1958.
[61] Wu H, Gu X. Max-pooling dropout for regularization of convolutional neural networks. In: Neural Information Processing: 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9–12, 2015, Proceedings, Part I 22. Springer; 2015. p. 46–54.
[62] Garbin C, Zhu X, Marques O. Dropout vs. batch normalization: an empirical study of their impact to deep learning. Multimed Tools Appl. 2020;79(19):12777–12815. doi: 10.1007/s11042-019-08453-9
[63] Ko B, Kim HG, Oh KJ, et al. Controlled dropout: a different approach to using dropout on deep neural network. In: 2017 IEEE International

Conference on Big Data and Smart Computing (BigComp). IEEE; 2017. p. 358–362.
[64] Skourt BA, El Hassani A, Majda A. Mixed-pooling-dropout for convolutional neural network regularization. J King Saud Univ Comput Inf Sci. 2022;34(8):4756–4762.
[65] Khan SH, Hayat M, Porikli F. Regularization of deep neural networks with spectral dropout. Neural Netw. 2019;110:82–90. doi: 10.1016/j.neunet.2018.09.009
[66] Shirke V, Walika R, Tambade L. Drop: a simple way to prevent neural network by overfitting. Int J Res Eng Sci Manag. 2018;1:2581–5782.
[67] Hou W, Wang W, Liu RZ, et al. Cropout: a general mechanism for reducing overfitting on convolutional neural networks. In: 2019 International Joint Conference on Neural Networks (IJCNN). IEEE; 2019. p. 1–8.
[68] Poernomo A, Kang DK. Biased dropout and crossmap dropout: learning towards effective dropout regularization in convolutional neural network. Neural Netw. 2018;104:60–67. doi: 10.1016/j.neunet.2018.03.016
[69] Bieder F, Sandkühler R, Cattin PC. Comparison of methods generalizing max-and average-pooling. Preprint; 2021. arXiv:210301746.
[70] Zafar A, Aamir M, Mohd Nawi N, et al. A comparison of pooling methods for convolutional neural networks. Appl Sci. 2022;12(17):8643. doi: 10.3390/app12178643
[71] Chollet F. Deep learning with python. Shelter Island: Simon and Schuster; 2021.
[72] Han J, Kamber M, Pei J. 2 – getting to know your data. In: Han J, Kamber M, Pei J, editors. The Morgan Kaufmann series in data management systems. Boston: Morgan Kaufmann; 2012. p. 39–82. doi: 10.1016/B978-0-12-381479-1.00002-2
[73] Japkowicz N, Shah M. Evaluating learning algorithms: a classification perspective. Cambridge: Cambridge University Press; 2011.
[74] Ying X. An overview of overfitting and its solutions. In: Journal of Physics: Conference Series. Vol. 1168. IOP Publishing; 2019. p. 022022.
[75] Brigato L, Mougiakakou S. No data augmentation? Alternative regularizations for effective training on small datasets. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; 2023. p. 139–148.
[76] Prechelt L. Early stopping-but when? In: Neural networks: tricks of the trade. Heidelberg: Springer; 2002. p. 55–69.
[77] Ba J, Frey B. Adaptive dropout for training deep neural networks. Adv Neural Inf Process Syst. 2013;26.
[78] Cai S, Shu Y, Chen G, et al. Effective and efficient dropout for deep convolutional neural networks. Preprint; 2019. arXiv:190403392.
[79] Furusho Y, Ikeda K. Resnet and batch-normalization improve data separability. In: Asian Conference on Machine Learning. PMLR; 2019. p. 94–108.
[80] Murray N, Perronnin F. Generalized max pooling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2014. p. 2473–2480.
[81] Momeny M, Jahanbakhshi A, Jafarnezhad K, et al. Accurate classification of cherry fruit using deep CNN based on hybrid pooling approach. Postharvest Biol Technol. 2020;166:111204. doi: 10.1016/j.postharvbio.2020.111204
[82] Lehmann MK, Nguyen U, Allan M, et al. Colour classification of 1486 lakes across a wide range of optical water types. Remote Sens. 2018;10(8):1273. doi: 10.3390/rs10081273
[83] Hu H. Research on colour recognition sorting method of waste plastic bottles based on computer perspective. In: 2022 4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM). IEEE; 2022. p. 914–918.
[84] Petus C, Waterhouse J, Tracey D, et al. Using optical water-type classification in data-poor water quality assessment: a case study in the torres strait. Remote Sens. 2022;14(9):2212. doi: 10.3390/rs14092212
[85] Wei X, Bohrer B, Uttaro B, et al. Centre pork chop colour classification using image analysis on the ventral surface of the loin. Can J Anim Sci. 2023:123–126.
[86] Pegalajar M, Ruiz L, Criado-Ramón D. Munsell soil colour classification using smartphones through a neuro-based multiclass solution. AgriEngineering. 2023;5(1):355–368. doi: 10.3390/agriengineering5010023
[87] Reyes JF, Contreras E, Correa C, et al. Image analysis of real-time classification of cherry fruit from colour features. J Agric Eng. 2021;52(4). doi: 10.4081/jae.2021.1160
[88] Göksel Duru D, Alobaidi M. Classification of brain electrophysiological changes in response to colour stimuli. Phys Eng Sci Med. 2021;44(3):727–743. doi: 10.1007/s13246-021-01021-2
[89] van Minderhout HM, Joosse MV, Grootendorst DC, et al. Eye colour and skin pigmentation as significant factors for refractive outcome and residual accommodation in hypermetropic children: a randomized clinical trial using cyclopentolate 1% and tropicamide 1%. Acta Ophthalmol. 2022;100(4):454–461. doi: 10.1111/aos.v100.4
[90] Sukhetha P, Hemalatha N, Sukumar R. Classification of fruits and vegetables using ResNet model. agriRxiv; 2021. 20210317,450.
[91] Gouda N, Amudha J. Skin cancer classification using ResNet. In: 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA); 2020. p. 536–541. doi: 10.1109/ICCCA49541
[92] Singh A, Bay A, Mirabile A. Assessing the importance of colours for CNNs in object recognition. Preprint; 2020. arXiv:201206917.
[93] Mathew MB, Surya Manjunathan G, Gokul B, et al. Banana ripeness identification and classification using hybrid models with ResNet-50, VGG-16 and machine learning techniques. In: Machine Intelligence Techniques for Data Analysis and Signal Processing: Proceedings of the 4th International Conference MISP 2022. Vol. 1. Springer; 2023. p. 259–273.
[94] Zhang C, Xia K, Feng H, et al. Tree species classification using deep learning and RGB optical images obtained by an unmanned aerial vehicle. J For Res. 2021;32(5):1879–1888. doi: 10.1007/s11676-020-01245-0
[95] Reddy SR, Varma GS, Davuluri RL. Resnet-based modified red deer optimization with DLCNN classifier for plant disease identification and classification. Comput Electr Eng. 2023;105:108492. doi: 10.1016/j.compeleceng.2022.108492