VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

Investigation of Generalization Properties of Convolutional Neural Networks

Konvoliucinių neuroninių tinklų generalizavimo savybių tyrimas

Master`s final thesis

Made by: Edvinas Ščigla _____
(signature)

Supervisor: Dr. Olga Kurasova _____
(signature)

Reviewer: Dr. Virginijus Marcinkevičius_____
(signature)

Vilnius, 2023

Summary

In this master's thesis the analysis of the current state of generalization properties on convolutional neural networks was done, investigated the impact of plain convolutional neural network architecture (ResNet and DenseNet with different number of layers) on convolutional neural network training and generalization and compared with ensembled network models. Investigated the impact of data augmentation on image level (CIFAR-10 and CIFAR-100 datasets), implemented data augmentation on non-image level and tested what impact it has to generalization compared with image level results. Conducted experiments allowed us to draw a conclusion that: data augmentation on non-image level brings better generalization compared to traditional data augmentation on image level. Both ensembled network models (ResNet and DenseNet) showed better results on generalization compared with plain convolutional neural network architectures, because individually each convolutional neural network has its own weaknesses, but together when they aggregated to generate single output – they are showing slightly better generalization results.

Keywords: generalization, convolutional neural networks, neural network architecture, data augmentation on image level, data augmentation on non-image level, ensembled networks.

Santrauka

Šiame magistro baigiamajame darbe yra pateikta dabartinė konvoliucinių neuroninių tinklų generalizavimo savybių analizė, ištirtas konvoliucinio neuroninio tinklo architektūros (ResNet ir DenseNet su skirtingu sluoksnių skaičiumi) poveikis konvoliucinio neuroninio tinklo mokymui ir generalizavimui, bei palyginti rezultatai su "sujungtais" tinklo modeliais. Abu "sujungti" tinklo modeliai (ResNet ir DenseNet) parodė geresnius generalizavimo rezultatus, palyginus su įprastomis konvoliuciniais neuroniais tinklų architektūromis, nes kai keli modeliai yra sujungiami, kad būtų sukurta viena išvestis - jie rodo šiek tiek geresnius generalizavimo rezultatus. Taip pat buvo palyginta duomenų augmentacijos įtaką generalizavimui (buvo naudojami CIFAR-10 ir CIFAR-100 duomenų rinkiniai). Augmentacija buvo pritaikyta visam vaizdui iš karto, taip pat skaidant vaizdą į kelias dalis. Atlikti eksperimentai leido padaryti išvadą, kad: duomenų augmentavimas ne vaizdo lygiu suteikia geresnį generalizavimą, palyginus su tradicine duomenų augmentacija (kai augmentacija taikoma visam vaizdui iš karto).

Raktiniai žodžiai: generalizavimas, konvoliuciniai neuroniniai tinklai, neuroninio tinklo architektūra, viso vaizdo duomenų augmentacija, vaizdo skaldymas ir duomenų augmentacija kiekvienai išskaldytai daliai, sujungti tinklai.

Content

1. Introduction

Deep neural networks are one of the trends in the development of artificial intelligence systems. The idea behind this concept is like the human nervous system - namely, its ability to learn, also capability to act based on previous experience, making fewer mistakes over time. The power of neural network depends on how well it describes new data after completing the training process [Hay98]. Generalization is a measure of how well the deep neural network performs on the actual problem once training is complete.

Relevance of the Topic

In recent years, the deep convolutional neural network (CNN) has made remarkable achievements in computer vision, including image classification [LZ16], however, its huge structure and massive parameters pose a challenge to the training of the network and can easily overfit a training set with numerous images leading to a large generalization gap on test and new data [XCW+21]. The generalization of convolutional neural models refers to their ability to adapt to the new, previously unseen data that come from the same distribution as that used when the model was learned. Data augmentation has been proposed to solve this problem, but usually it is performed at the image level, however it is rarely studied what impact to generalization have data augmentation on non-image level [HFL+22]. Generalization ability strongly depends on the relation between the size of learning data and the complexity of network architecture. CNN architecture might be critical factor in improving generalization, so the choice what model of CNN to select is very important, furthermore different manipulations of architecture decisions might lead to better results when comparing plain CNN model with modified on generalization impact.

Aim of the Final Project

The aim of the master thesis is to explore the impact of network architecture and data augmentation on image and non-image levels on convolutional neural network training and generalization.

Objectives of the Final Project

1) Analysis of the current state of generalization properties on convolutional neural networks.
2) Investigate impact of plain convolutional neural network architecture on convolutional neural network training and generalization.
3) Investigate impact of ensembled convolutional neural network models compared with plain convolutional neural networks model on convolutional neural network training and generalization.
4) Investigate impact of data augmentation on image level on convolutional neural network training and generalization.
5) Investigate impact for convolutional neural network training and generalization of data augmentation at non-image level and compare with the results that retrieved from previous point.
6) To make conclusions according to the research about generalization properties

Expected results

1) Examined impact of convolutional neural network architecture on convolutional neural network training and generalization.

2) Examined impact of ensembled models compared with plain CNN models on training and generalization.

3) Examined impact of data augmentation on image level on convolutional neural network training and generalization.

4) Examined impact of data augmentation of non-image level and compared with impact for image level augmentation on convolutional neural network training and generalization.

2. Analysis of scientific literature

In this section, we will briefly review the main sources of literature related to generalization properties of convolutional neural networks. In subsections 1.1 and 1.2 will be shortly presented biological model of neuron and his artificial model called single-layer perceptron. In 1.3 subsection will be described artificial neural network in general and in 1.4 subsection will briefly present convolutional neural network, one of the popular classes of artificial neural network. After general introduction will be dived to investigation of convolutional neural network and generalization properties of it:

1. Training dataset impact, different problems that might occur and how they can influence the performance of convolutional neural network.
2. Convolutional neural network architectures (1 classic and 2 modern), their differences, how they work and what impact they have to generalization.
3. Learning rate impact, common methods for learning rate schedule, also reviewing other methods that might have impact for learning rate.

2.1. Biological neuron model

The cortex of the human brain is covered with a layer of 2-3 millimeters thick, which consists of neurons. A neuron is the no-frills component of the brain. It is a specific cell consisting of four elements: axons, dendrites, soma and synapse (Fig. 1). Through the dendrites, inputs from the environment that are combined in the soma come to the neural soma, operations are performed in axons, and the transmission of output through the synapses occurs.

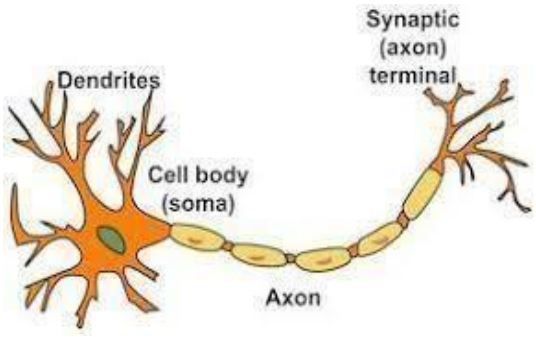

Figure 1. Biological neuron [Pra21].

The neurons themselves are relatively slow acting, with a speed of processing and transmitting information of about 80-120ms [SS06]. However, there are so many neurons in the brain and their connections that their slow operation does not cause additional problems.

2.2. Artificial neuron model (single-layer perceptron)

The simplest model of an artificial neural network is a single-layer perceptron. It can consist of one or more parallel connected artificial neurons, the outputs of which are fed into the functions of the threshold. A perceptron is an iteratively trained linear classifier that consists of x = $\{x_0, x_1, x_2, ..., x_p\}$ of the training set of vectors called inputs, $\{w_0, w_1, w_2, ..., w_p\} \in \mathbb{R}$ transfer coefficients called weights, activation (transmission) function f (a) and $\{y_1, y_2, ..., y_n\}$ values called outputs. The input x_0 is called zero input and its value is constant $x_0 = 1$, and w_0 is called zero weight or threshold (bias). The perceptron is shown in Figure 2.

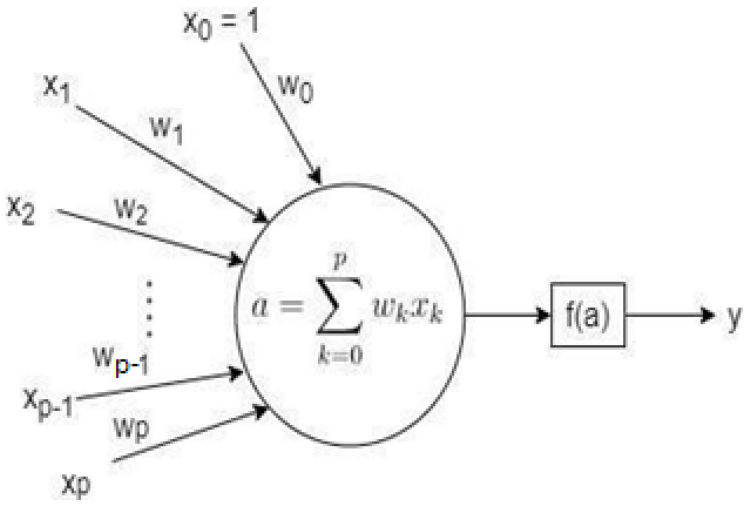

Figure 2. Perceptron, adapted from [Fuk80].

For the perceptron to solve the task of specific classification, it must be trained. Perceptron training is an iterative process in which the weights W = $\{w_0, w_1, w_2, ..., w_p\}$ are found, with which the result of the function acquires the lowest possible value.

2.3. Artificial neural networks

Artificial neural networks are connected networks with tops of perceptrons, and the oriented edges of which combine 2 perceptrons, of which the output of one perceptron is used as the entrance of another perceptron. According to the structure of the network, artificial neural networks are divided into feed-forward and feedback. Feedback artificial neural networks have at least one cycle and feed-forward neural networks do not. Cycles are used in neural network models to try to simulate the human brain, feedback cycles (loops) allow models to know what they did right or wrong, so that it may continue to learn from this data to perform better the next time. Feed-forward neural networks, due to their simplicity, require a shorter training time than feed-back neural networks. This is one of the reasons for the increased popularity of neural networks of direct propagation according to Murat Hüsnü [Hus06]. Feed-forward networks are grouped into single layer perceptrons, multi-layer perceptrons and networks of radial basis function. This work study convolutional neural networks that are extensions of multilayer perceptrons.

Multi-layer perceptron (MLP) is an artificial neural network whose perceptrons are grouped into layers that may contain a different number of perceptrons. The perceptrons in each layer have the same activation function. The layers of the multilayer perceptron are arranged in a row and the outputs of all perceptrons in each layer are the entrances of all perceptrons of the further layer. The first layer is called the input layer and it consists not of perceptrons, but of $\boldsymbol{x}$ components of the teaching data vector. The last layer is the output layer and contains as many perceptrons as there are classes under consideration, each class is assigned a preceptron corresponding to it. The output values of the output layer perceptrons depend on the activation function used in them. The remaining layers are called hidden layers. The multi-layer perceptron is shown in Figure 3.

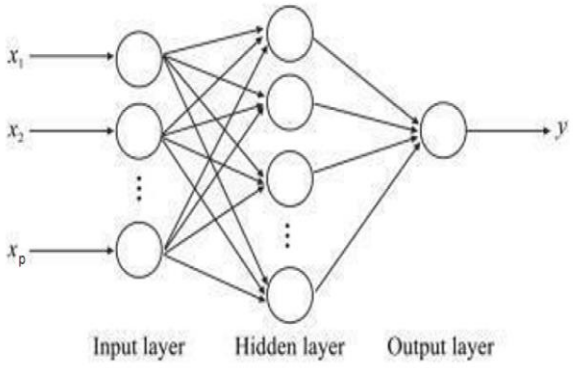

Figure 3. Multi-layer perceptron (MLP), adapted from [Fuk80].

For a multi-layer perceptron to perform classification, it must be trained. Like perceptron, multi-layer perceptron is trained in iteratively changing the weights of all perceptrons.

Radial basis functions are a special class of feed-forward neural networks consisting of three layers: an input layer, a hidden layer, and the output layer. This is fundamentally different from most neural network architectures, which are composed of many layers and bring about nonlinearity by recurrently applying non-linear activation functions. The input layer receives input data and passes it into the hidden layer, where the computation occurs. The hidden layer of radial basis functions neural network is the most powerful and very different from most neural networks. The output layer is designated for prediction tasks like classification or regression.

A radial basis function (shortly RBF) is a real-valued function, the value of which depends only on the distance from the origin. Although we use various types of radial basis functions, the Gaussian function [Guo11] is the most common. In the instance of more than one predictor variable, the radial basis functions neural network has the same number of dimensions as there are variables. If three neurons are in a space with two predictor variables, we can predict the value from the RBF functions. We can calculate the best-predicted value for the new point by adding the output values of the RBF functions multiplied by the weights processed for each neuron. The radial basis function for a neuron consists of a center and a radius (also called the spread). The radius may vary between different neurons. Multi-layer perceptron and radial basis Function (RBF) are popular neural network architectures called feed-forward networks. The main differences between RBF and MLP are that multi-layer perceptron consists of one or

several hidden layers, while Radial Basis function consists of just one hidden layer. RBF network has a faster learning speed compared to MLP. In MLP, training is usually done through back-propagation for every layer, but in RBF, training can be done either through back-propagation or RBF network hybrid learning.

2.4. Convolutional neural networks

Convolutional neural network is a type of artificial neural network most used in the field of computational vision. The initial structure and functioning of convolutional neural networks were inspired by the imaging processes taking place in the brain, however, the technical implementation of this method in nature and computers differs fundamentally according Kunihiko Fukushima [Fuk80].

Convolutional neural networks are among the most popular deep neural networks. A convolutional neural network is a deep neural network in which a convolution operation is used in at least one layer. The first successful implementation of the convolutional neural network is described in the work [LBD+89], this network is designed to recognize handwritten postcodes. Convolution is a mathematical operation on two functions (f and g) that produces a third function *(f * g)* that expresses how the shape of one is modified by the other. The term convolution refers to both the result function and to the process of computing it. It is defined as the integral of the product of the two functions after one is reversed and shifted. The integral is evaluated for all values of shift, producing the convolution function:

$$(f * g)(t) = \int_a^b f(\tau)\, \mathrm{g}(\mathrm{t} - \tau)\mathrm{d}\tau$$

While the symbol t is used above, it need not represent the time domain, at each t, the convolution formula can be described as the area under the function f(τ) weighted by the function g(−τ) shifted by the amount t. As t changes, the weighting function g(t − τ) emphasizes different parts of the input function f(τ). If t is a positive value, then g(t − τ) is equal to g(−τ) that slides or is shifted along the τ-axis toward the right (toward b) by the amount of *t*, while if t is a negative value, then g(t − τ) is equal to g(−τ) that slides or is shifted toward the left (toward a) by the

amount of |t|.

Due to sparse interaction convolutional neural networks consider only significant subdivisions. Therefore, the training of convoluted neural networks takes less time and noise has less influence on the result.

The use of convolution in deep neural networks improves the learning process due to the principles of convolution: sparsity, parameter sharing and equivalent rendering.

Sparsity of connections means that some parameters are simply missing, nothing to do with sharing same non-zero parameter. Parameters in this case are zero, ignored. That means that not necessarily all (potential) inputs to a layer are connected to that layer, only some of them, rest are ignored.

In a simple multi-layer perceptron, all perceptrons in each layer have one connection to each perceptron of the further layer. At that time sparse interactions/connectivity are applied in convolutional networks. Sparse interaction is the result of a convolution on an artificial neural network, which leads to a decrease in the number of connections with the perceptrons of the further layer in the layers where the convolution is applied. An example of the sparse interaction between the two layers is represented in Figure 4.

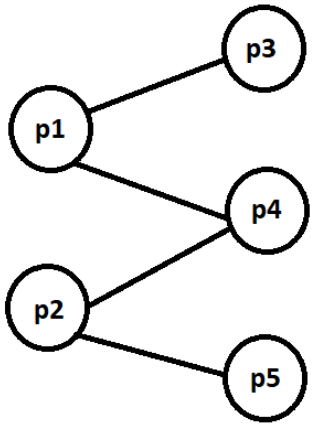

Figure 4. Sparse interaction.

Due to sparse interaction convolutional neural networks consider only significant subdivisions. Therefore, the training of convoluted neural networks takes less time and noise has less influence on the result.

Another reason why convolutional neural networks are faster than multi-layered perceptrons is parameters division. The sharing of parameters is the principle by which at least one parameter is used for more than one function. In the case of convolutional neural networks, each member of the nucleus is used for each input member. At that time, in a multilayer perceptron, the weights of which are matrices, each member of the weight matrix is used for only one member of the input. Therefore, weights take up less space in the memory of the computer (Fig. 5).

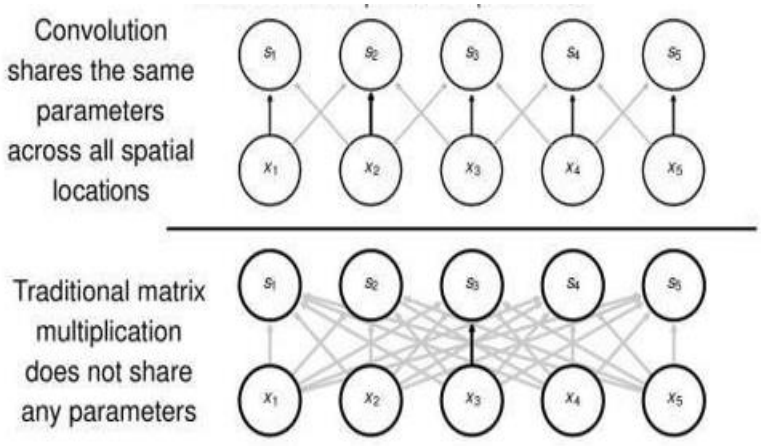

Figure 5. Parameter sharing [CSW+19].

The last principle is equivalence. Function f(x) is equivalent to function g(x) if equality f(g(x)) = g(f(x)) = g(f(x)) is satisfied. Convolution is the equivalent of many functions of matrix translation. The benefits of this principle are manifested if the available function of a small number of neighboring pixels is useful when it is applied in many places of input.

2.5. Impact of training dataset on CNN training and generalization

The purpose of training a convolutional neural network (CNN) is to obtain weight factors that give high classification accuracies. Data set is frequently emphasized factor in the pantheon of factors that affect artificial neural network generalization ability, because it is the first thing that network receive as an input. In this section it will be reviewed common problems that related to datasets and how to solve them.

Overfitting occurs when the accuracy of your training dataset, the dataset used to train the network, is greater than your testing accuracy [CAG+15]. Also need to mention that if a model has been trained too well on training data, it will be unable to generalize, it will make inaccurate predictions when given new data, making the model useless even though it is able to make accurate predictionsfor the training data. Also, overfitting reveals itself when your network has a low error rate in the training set and a higher error rate in the testing set. One of the main reasons for the network to overfit is if the size of the training dataset is small according to Pavlo Radiuk [Rad17]. When the network tries to learn from a small dataset it will tend to have greater control over the dataset and will make sure to satisfy all the datapoints exactly. It can be thought of as the network trying to memorize every single data point failing to capture the general trend in the data.

To improve generalization performance, many explicit and implicit regularization techniques are proposed, such as: early stopping, weight decay, data augmentation and dropout.

Dropout is a technique to prevent neural networks from overfitting. Dropout works by randomly disabling neurons and their corresponding connections. Is implemented per-layer in a neural network, may be implemented on any or all hidden layers in the network as well as the visible or input layer, it is not used on the output layer (Fig. 6).

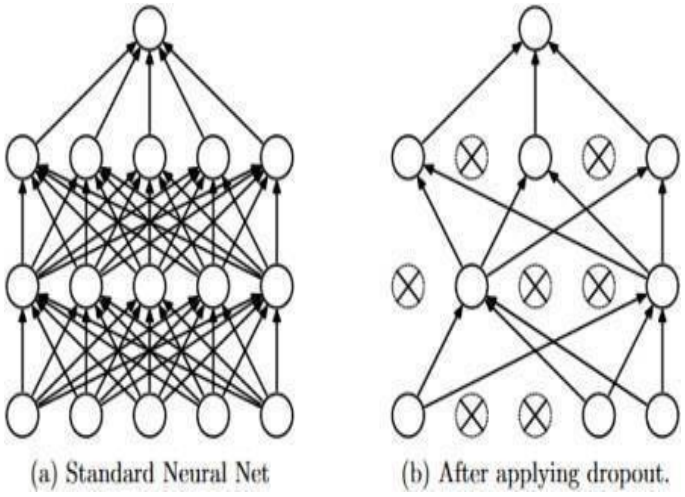

Figure 6. Dropout regularization [CSW+19].

At the left – standard neural network with 2 hidden layers. At the right – an example of a thinned network produced by applying dropout to the network on the left. Crossed units have been dropped. Dropout regularization to convolutional neural networks fail to obtain noticeable performance improvement according Shaofeng Cai research [CSW+19]. The effectiveness of dropout for CNNs is further reduced by the introduction of other regularization techniques such as data augmentation and batch normalization.

Data augmentation is a technique to artificially create new training data from existing training data. This is done by applying domain-specific techniques to examples from the training data that create new and different training examples. Image data augmentation is perhaps the most well-known type of data augmentation and involves creating transformed versions of images in the training dataset that belong to the same class as the original image. Transforms include a range of operations from the field of image manipulation, such as shifts, flips, zooms, and much more (Fig. 7).

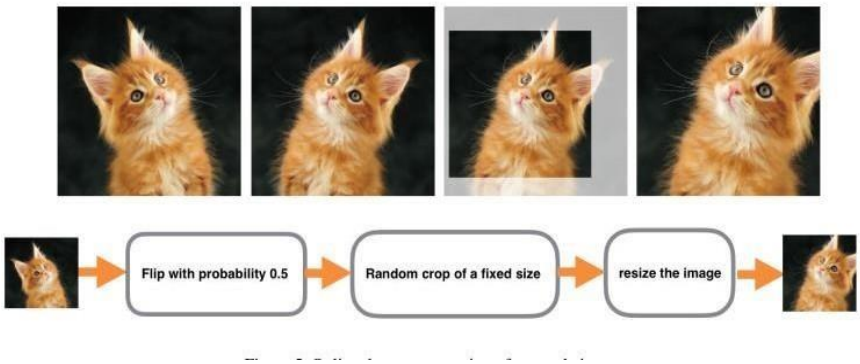

Figure 7. Data augmentation [EA21].

According to Marcelo Romero Aquino [AGH+17] small augmentation sizes do not have much influence on the performance of the classifiers for both, balanced and unbalanced datasets, but in another hand - a high augmentation size does not seem to induce overfitting: instead, it allows to improve the generalization capability of the model. Two different strategies were applied for received research output: only augmented (this strategy consisted in augmenting the samples at each epoch of the training phase without considering if it is balanced or not) and balanced augmented (this strategy consists in balancing the train set before starting to train the classifier and applying on-line augmentation). The balanced augmented strategy allows to obtain

an improvement of the performance with a smaller augmentation size than the only augmented strategy for an unbalanced dataset.

Batch Normalization is a normalization technique done between the layers of a neural network instead of in the raw data. It is done in mini batches instead of the full data set. It serves to speed up training and use higher learning rates, making learning easier (Fig. 8).

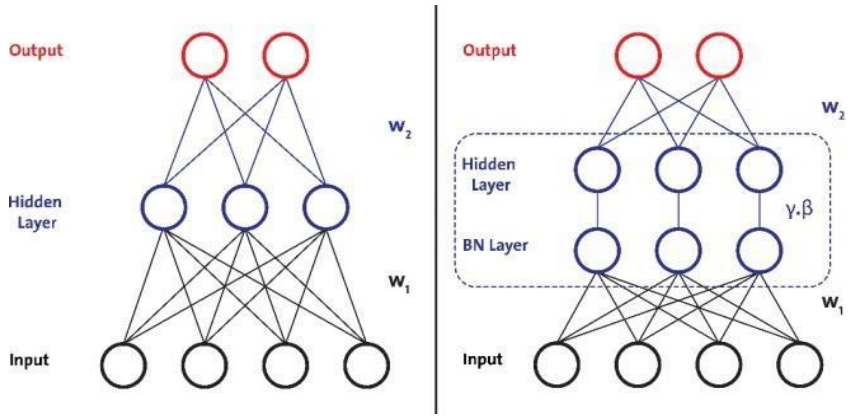

Figure 8. Batch normalization [SKP18].

It is a layer that allows every layer of the network to learn more independently. It is used to normalize the output of the previous layers. The activations scale the input layer in normalization. Using batch normalization learning becomes efficient also it can be used as regularization to avoid overfitting of the model. The layer is added to the sequential model to standardize the input or the outputs. It can be used at several points in between the layers of the model. It is often placed just after defining the sequential model and after the convolutional and pooling layers. Since the means and variances are fixed during inference, the normalization is simply a linear transform applied to each activation. It may further be composed with the scaling by γ and shift by β, to yield a single linear transform [IS15]. According to Shiro Takagi and Yuki Yoshida [TY20] batch normalization turned out to improve the efficiency of CNN training significantly. It helps avoid vanishing or exploding gradient by keeping the activations more stable during training, which in turn allows to use larger learning rates. It also makes the training less dependent on the initialization and works as a regulator that in some cases eliminates the need for dropout.

Batch normalization can be applied after the convolution and before the nonlinear

activation function. When the convolution has multiple output channels, we need to carryout batch normalization for each of the outputs of these channels, and each channel has its own scale and shift parameters, both of which are scalars. Assume that our minibatches contain *m* examples and that for each channel, the output of the convolution has height *p* and width *q*. For convolutional layers, we carry out each batch normalization over the $m \times p \times q$ elements per output channel simultaneously. Thus, we collect the values over all spatial locations when computing the mean and variance and consequently apply the same mean and variance within a given channel to normalizethe value at each spatial location. Batch normalization typically behaves differently in training mode and prediction mode according Santurkar [STI+18], because after training, we use the entire dataset to compute stable estimates of the variable statistics and then fix them at prediction time, consequently, batch normalization behaves differently during training and at test time. Recall that dropouts also exhibit this characteristic.

But according to that there are many situations under which batch normalization starts to hurt performance or does not work at all. In *"understanding regularization in batch normalization"* research work writers Ping Luo and Wenqi Shao investigate how batch normalization works with small size batches [LS18]. The batch normalization layer must calculate mean and variance to normalize the previous outputs across the batch. This statistical estimation will be pretty accurateif the batch size is large while keeps on decreasing as the batch size decreases (Fig. 9).

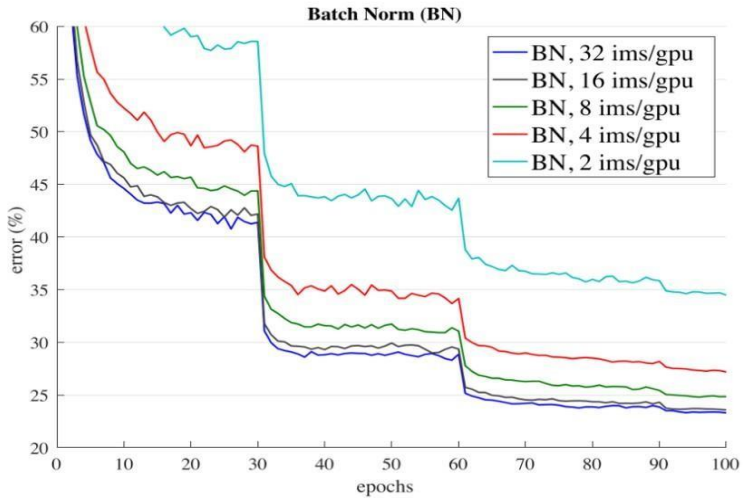

Figure 9. Batch normalization [GG17].

From Figure 9, if the batch size is kept 32, it is final validation error is around 23 and the error keeps on decreasing with smaller batch sizes (batch size cannot be 1 for batch normalization because it will be mean of itself), and there is a huge difference in the loss, around 10 percent. Also, in research batch normalization leads to increased training time, it happens because batch norm requires double iteration through input data, one for computing batch statistics and another for normalizing the output.

Another issue with data set and training is Underfitting. Underfitting occurs when there is still room for improvement on the train data. A network is said to be underfitting when it is not able to classify the data it was trained on. If the network is unable to classify data it was trained on, it is likely not going to do well at prediction data that it has not seen before (Fig. 10).

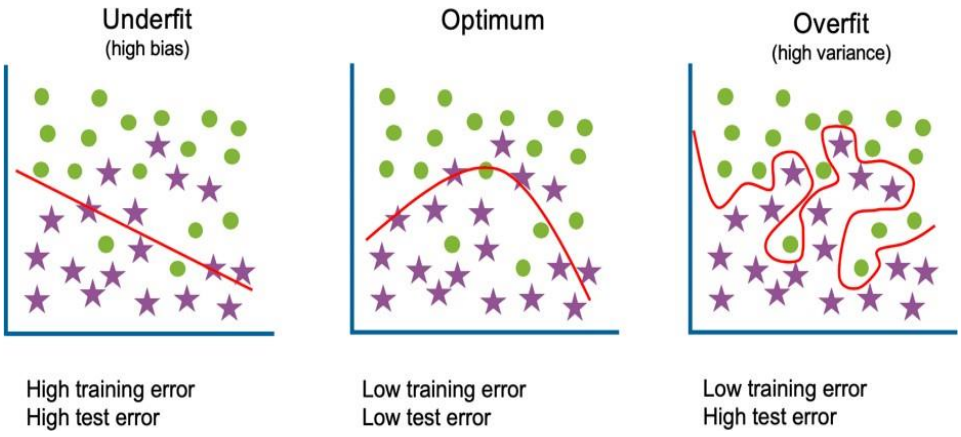

Figure 10. Examples of underfitting and overfitting [EA21].

Some very common approaches to address the underfitting issue are to increase the dataset and train for a longer time, to check for a more proper regularization, because by decreasing the amount of regularization, more complexity and variation is introduced into the model, allowing for successful training of the model. Another thing to try is to check if the model is not powerful enough (the architecture needs to change), but another problem might occur here, because using popular convolutional neural network architectures, which contain many convolutional layers with potentially hundreds to millions of trainable parameters, the probability of overfitting is very high, solution for this problem is to increase the flexibility of the model, which may be achieved with regularization techniques like previously mentioned: dropout method and batch normalization.

2.6. Impact of network architecture on convolutional neural network training and generalization

Convolutional neural network architecture is formed by a stack of distinct layers that transform the input volume into an output volume (as example: holding the class scores) through a differentiable function. CNNs are like other neural networks, but they have an added layer of complexity since they use a series of convolutional layers. Convolutional layers are an essential component of Convolutional Neural Networks. A convolutional neural network architecture usually consists of a couple of convolutional layers, each of which is followed by a pooling layer. These layers have the purpose of extracting features and shrinking the dimensionality of the output. The picture below represents a typical CNN architecture (Fig. 11).

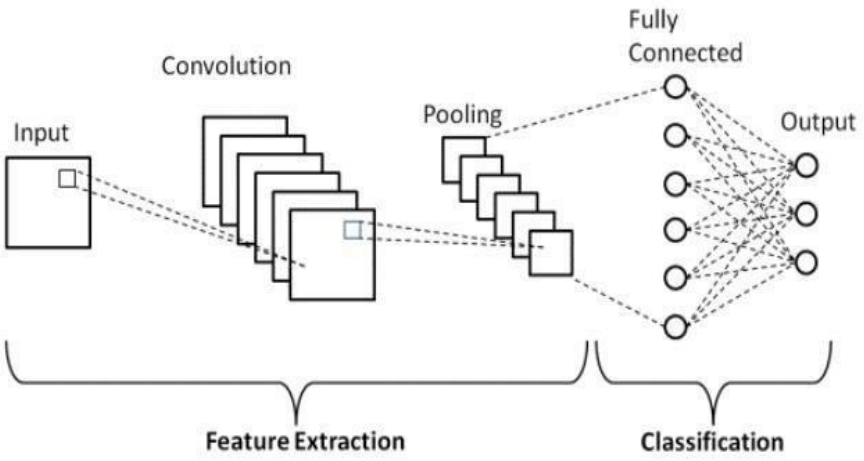

Figure 11. CNN architecture [CSW+19].

Convolutional layers are made up of a set of filters (also called kernels) that are applied to an input image. The output of the convolutional layer is a feature map, which is a representation of the input image with the filters applied. Convolutional layers can be stacked to create more complex models, which can learn more intricate features from images.

Pooling layers are a type of layer used in deep learning. Pooling layers reduce the spatial size of the input, making it easier to process and requiring less memory. Pooling also helps to reduce the number of parameters and makes training faster. There are two main types of pooling:

max pooling and average pooling. Max pooling takes the maximum value from each feature map, while average pooling takes the average value. Pooling layers are typically used after convolutional layers to reduce the size of the input before it is fed into a fully connected layer.

Fully connected layers are one of the most basic types of layers in a convolutional neural network. As the name suggests, each neuron is fully connected to every other neuron in the previous layer. Fully connected layers are typically used towards the end of a CNN, when the goal is to take the features learned by the previous layers and use them to make predictions. For example, if we were using CNN to classify images of animals, the final fully connected layer might take the features learned by the previous layers and use them to classify an image of what animal appears in picture.

Over the years, variants of CNN architectures have been developed, leading to amazing advances in the field of deep learning. Deep dive some of the popular CNN architectures that stood out in their approach and significantly improved on the error rates as compared to their predecessors:

LeNet-5 [LBB+98] architecture is perhaps the most widely known CNN architecture. It was created by Yann LeCun in 1998 and widely used for written digits recognition. LeNet- 5 was trained on 2D images, grayscale images with a size of 32×32×1 (Fig. 12).

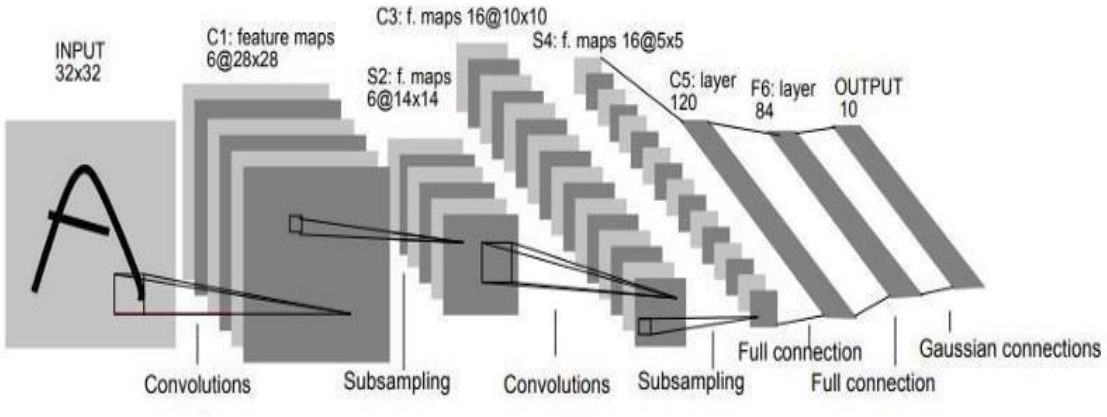

Figure 12. LeNet-5 architecture and training [LBB+98].

The LeNet-5 architecture consists of 3 convolutional layers, 2 subsampling layers and 2 fully connected layers. The subsampling layer implements average pooling to reduce the dimensions of image, thus reducing the computational complexity of the image while reducing the dependency of the model on the location of the features instead of their shape. Average pooling involves calculating the average for each patch of the feature map. This means that each 2×2 square of the feature map is down sampled to the average value in the square.

The first layer is the input layer, this is generally not counted as a layer of the network as nothing is learnt in this layer. The input layer is built to take in 32×32 images. The grayscale images used in the research paper [LBB+98] were normalized to have a mean of 0 and a standard deviation of 1. The benefits of normalizing images can be seen in reduced training times.

Layer *C1* (Fig. 12) is a convolutional layer with six 5×5 convolution kernels, and the feature allocation size is 28×28 (because *MNIST* dataset are 28×28 in dimensions), whereby input image information can be avoided.

Layer *S2* (Fig. 12) is the under sampling / grouping layer which generates 6 function graphs of length 14×14. Each cell in every function map is attached to 2×2 neighborhoods at the corresponding function map in *C1*. Layer *S4* (fig. 13) is just like *S2* with a length of 2×2 and an output of sixteen 5×5 function graphics.

C3 convolution layer (Fig. 12) encompass sixteen 5×5 convolution kernels the input of the primary six function maps *C3* is every continuous subset of the 3 function maps in *S2*, the access of the following six function maps comes from the access of the 4 continuous subsets and the input for the following 3 function maps is crafted from the 4 discontinuous subsets. Finally, the input for the very last function diagram comes from all the *S2* function diagrams.

Layer *C5* (Fig. 12) is a convolution layer with one hundred twenty convolution cores of length 5x5. Each cell is attached to the 5×5 neighborhoods along sixteen S4 function charts. Since the function chart length of S4 is likewise 5×5, the output length of *C5* is 1×1, so *S4* and *C5* are absolutely linked. It is referred to as a convolutional layer in preference to a completely linked layer due to the fact if the input of LeNet-5 becomes large and its shape stays unchanged, then its output length is bigger than 1×1.

The *F6* layer (Fig. 12) is connected to *C5*, and 84 feature charts are generated. In the grayscale images used in the research, the pixel values from 0 to 255 were normalized to values between -0.1 and 1,175, the reason for normalization is to make sure the image stack has a mean of 0 and a standard deviation of 1.

The main disadvantage of LeNet-5 architecture according to Oleg Potkin [PP18] is overfitting in some cases and no built-in mechanism to avoid this. Also, this model was more specifically built for a certain use case. While it was a breakthrough in 1998, it does not do as well with color images. Most image recognition problems would require RGB images for better recognition.

In this work also reviewed most popular and newest (industry used) network architectures like: ResNet and DenseNet.

The DenseNet first appeared in 2016 in the paper called *"Densely Connected Convolutional Networks"* [HLM+17]. A DenseNet is a type of convolutional neural network that utilizes dense connections between layers, through Dense Blocks, where we connect all layers (with matching feature-map sizes) directly with each other (Fig. 13). To preserve the feed-forward nature, each layer obtains additional inputs from all preceding layers and passes on its own feature- maps to all subsequent layers. This allows later layers within the network to directly leverage the features from earlier layers, encouraging feature reuse within the network. According to Gao Huang [HLM+17] concatenating feature-maps learned by different layers increases variation in the input of subsequent layers and improves efficiency.

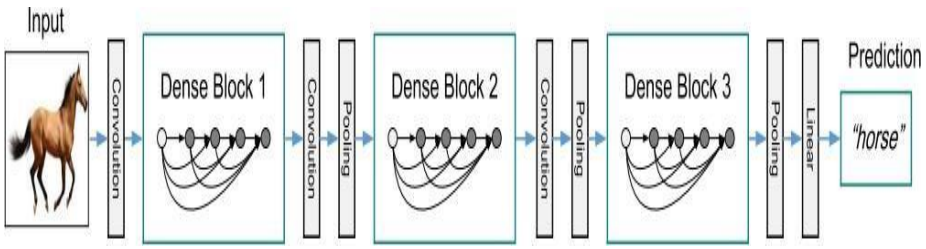

Figure 13. Dense blocks [HLM+17].

DenseNet architecture uses the same concepts of convolutions, pooling, but the important detail and innovation in this network architecture is the dense blocks. In these dense blocks, each layer takes all the preceding feature-maps as input, thus helping the training process by alleviating the vanishing-gradient problem. This vanishing gradient problem appears in deep networks where they are so deep that when we back-propagate the error into the network, this error is reduced at every step and eventually becomes 0. These connections basically allow the error to be propagated further without being reduced too much. These connections also encourage feature reuse and reduce the number of parameters, for the same reason, since it is reusing previous feature maps information instead of generating more parameters. And therefore, accessing the network "collective knowledge" and reducing the chance of overfitting, due to this reduction in total parameters.

Another modern architecture is ResNet. RestNet stands for residual network, it is an innovative neural network that was first introduced by Kaiming He in computer vision research paper titled "*Deep Residual Learning for Image Recognition*" [HZ15]. Mostly to solve a complex problem, we stack some additional layers in the deep neural networks which results in improved accuracy and performance. The intuition behind adding more layers is that these layers progressively learn more complex features, but a lot of researchers shows that with adding more layers on top of a network, its performance degrades [TAL16]. This problem of training very deep networks has been alleviated with the introduction of ResNet or residual networks and these ResNets are made up from residual blocks (Fig. 14).

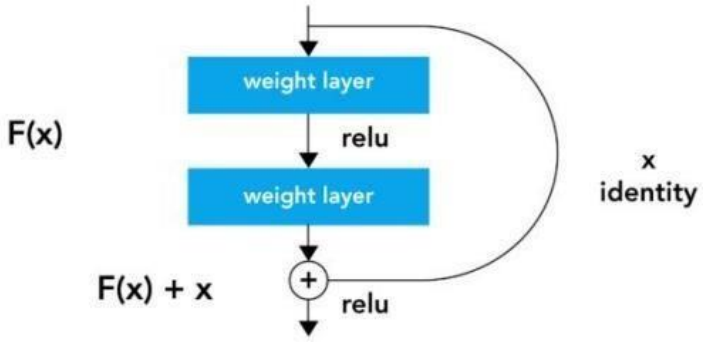

Figure 14. Residual learning: a building block [HZ15].

The very first thing we notice to be different is that there is a direct connection which skips some layers (may vary in different models) in between. This connection is called "*skip connection*" and is the core of residual blocks. Due to this skip connection, the output of the layer is not the same now.

Without using this skip connection, the input *x* gets multiplied by the weights of the layer followed by adding a bias term. Next, this term goes through the activation function, and we get our output. In another hand, with the introduction of skip connection, the output is changed to *x* provided to activation function (f(x)) plus x. The skip connections in ResNet solve the problem of vanishing gradient in deep neural networks by allowing this alternate shortcut path for the gradient to flow through.

However, the current ResNet has several architectural limitations in that residuals must be learned by fixed size shallow subnetworks, despite evidence that deeper networks are more expressive. Identity connections as implemented in the current ResNet also result in a mix of levels of feature representation at each layer, even though some features learned at earlier layers of a deep network may no longer provide useful information in later layers. A prior of the ResNet architecture is that learning identity weights is difficult, but by the same argument, it is difficult to learn the additive inverse of identity weights to remove information from the representation at any given layer.

According to Yuan Jiang and his research "*Exploring The Efficiency of Resnet and Densenet*" [Jia23], DenseNet has more advantages in generalization than ResNet, because DenseNet can retain more features in the cross-layer concatenate operation, which is more accurate than ResNet. The experimental results show that ResNet and DenseNet have good performance in the accuracy of gender classification on new testing data. Need to mention that only low number of layers were tested (18 and 12) and not only data augmentation was used (grayscale adjustment), but also image size normalization and image filtering. Author highlighting the importance of exploring the differences in performance between ResNet and DenseNet, uncovering the different advantages of Resnet and Densenet [Jia23].

In another paper called "*On the generalization effects of DenseNet model sctructures*" [LC18] by Yin Liu and Vincent Chen is written that modern neural network architectures take advantage of increasingly deeper layers, and various advances in their structure to achieve better generalization. In this experiment a low number of layers were also picked, because training was done with decreased samples count, so higher number of layers might produce severe gradient vanishing problems as the number of layers increases. Through experiments, the author concluded that certain neural network architectures contribute to their generalization abilities.

Conclusion of this network architecture impact on generalization is a good way to increase the generalization is the regularization of the architecture. It is implemented by the modification of structure, as well as using different methods of learning. It was shown that the explicit forms of regularization, such as weight decay, dropout, and even data augmentation, do not adequately explain the generalization ability of deep networks. The empirical observations [NBM+17] have shown that explicit regularization may improve the generalization performance of the network but is neither necessary nor by itself sufficient for controlling the generalization error.

2.7. Impact on convolutional neural network generalization ability by the size of the learning rate and learning rate schedule

The learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0. The learning rate controls how quickly the model is adapted to the problem and it is one of the most important hyperparameters for training neural networks. Thus, it is very important to set up its value as close to the optimal as possible. It is often helpful to decrease the learning rate over the course of training. The learning rate controls how big of a step for an optimizer to reach the minima of the loss function (Fig. 15).

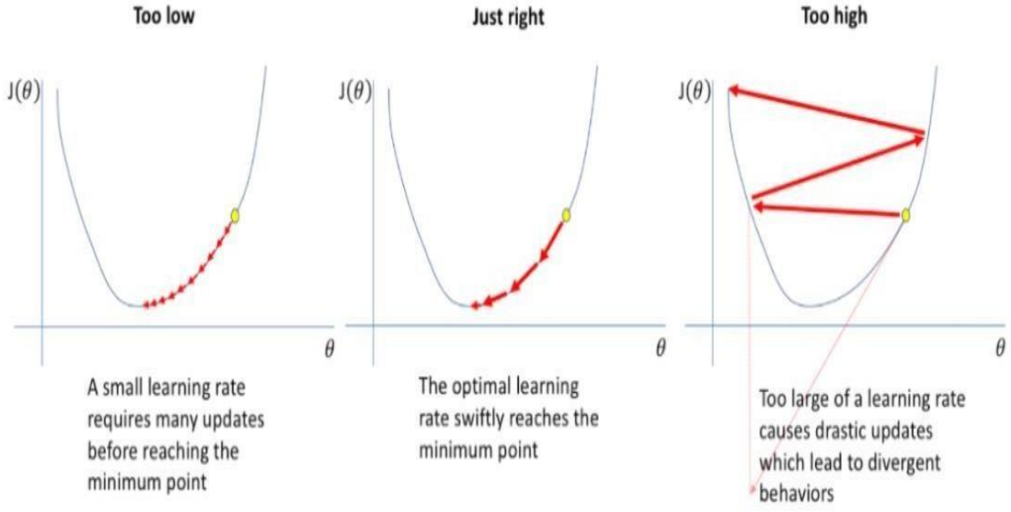

Figure 15. Learning Rates Cases [LZZ21].

With a large learning rate (on the right in 15 fig.), the algorithm learns fast, but it may also cause the algorithm to oscillate around or even jump over the minima. Even worse, a high learning rate equals large weight updates, which might cause the weights to overflow. On the contrary, with a small learning rate (on the left in fig. 15), updates to the weights are small, which will guide the optimizer gradually towards the minima. However, the optimizer may take too long to converge or get stuck in a plateau or undesirable local minima. A good learning rate is a tradeoff between the coverage rate and overshooting (in the middle in Figure 15). It's not too small so that ouralgorithm can converge swiftly, and it's not too large so that our algorithm would not jump back and forth without reaching the minima.

Although the theoretical principle of finding an appropriate learning rate is straightforward, but it is hard to achieve. To solve this problem, the learning rate schedule is introduced. A Learning rate schedule is a predefined framework that adjusts the learning rate between epochs or iterations as the training progresses. Two of the most common techniques for learning rate schedule are: constant learning rate and learning rate decay.

Constant learning rate – as the name suggests, we initialize a learning rate and do not change it during training. Learning rate decay is a technique for training modern neural networks. It starts training the network with a large learning rate and then slowly reducing/decaying it until local minima is obtained. It is empirically observed to help both

optimization and generalization. Since the learning rate is large initially, we still have relatively fast learning but as tending towards minima learning rate gets smaller and smaller, end up oscillating in a tighter region around minima rather than wandering far away from it. Learning rate decay can be described with formula (common method):

$$\alpha = \frac{1}{1+decayRate*epochNumber} * \alpha_0$$

where α is learning rate (current iteration), decayRate is hyper-parameter for the method is hyper-parameter for the method, α_0 is initial learning rate. Also, there are other methods to retrieve learning rate like: exponential decay, discrete staircase, epoch number based, mini-batch number based, manual decay.

Exponential decay can be described with formula: $\alpha = decayRate^{(epochNumber)} * \alpha_0$, this function applies an exponential decay function to a provided initial learning rate so that learning rate decay over time, exponentially. The decayRate of this method is always less than 1, the most used among practitioners is 0.95 according to Yang Li [LZZ21].

Discrete staircase - in this method learning rate is decreased in some discrete steps after every certain interval of time, for example you are reducing learning rate to its half after every 10 secs (Fig. 16).

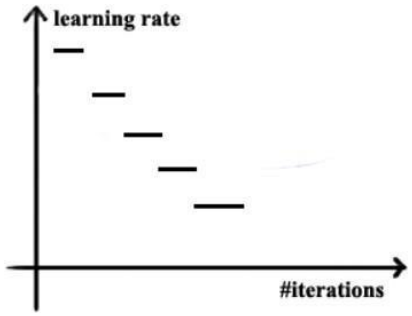

Figure 16. Discrete straicase in diagram [LZZ21].

Epoch number based - in this method we take some constant *k* and divide it with square root of epoch number. This can be described with formula: $\alpha = (k/\sqrt{epochNumber})* \alpha_0$.

Mini-batch number based - in this method we take some constant *k* and divide it with square root of Mini-batch number (*t* in formula) and this method is only used for Mini-batch gradient descent. This can be described with formula: $\alpha = (k/\sqrt{t})* \alpha_0$.

Manual decay - in this method practitioners manually examine the performance of algorithm and decrease the learning rate manually day by day or hour by hour.

In conclusion, as mentioned in many researches above – overfitting is common problem when talking about generalization in convolutional neural networks, because if model has been trained too well on training data, it will be unable to generalize, it will make inaccurate predictions when given new data, making the model useless even though it is able to make accurate predictions for the training data, so optimization and fixing the overfitting issue is absolutely necessary to obtain good generalization results. Also need to mention that learning rate is important, because it controls how quickly the model is adapted to the problem.

3. Related works

Most study cases are investigating data augmentation impact on generalization on image level [RGY+23], [SL20], [SM21], [LLL+21], but not on non-image levels (on patches). Latest work related with patches [HFL+22] have downside – applying random augmentation techniques can lead to corrupted image, because not at all images are fit to makes some kind of manipulations. In this work improvement implemented, several rules introduced that are applied to avoid receiving corrupted image, also investigated impact on generalization with comparison with traditional data augmentation on image level, that is missing in previous research. There is also multiply factors in network architecture, that can make impact on generalization, in researches [Jia23], [BSE+23] authors highlighting the importance of exploring the neural network architectures, differences in performance between ResNet and DenseNet, uncovering the different advantages of ResNet and DenseNet [Jia23]. When talking about CNN architecture, the most common investigations are done with plain network architectures [AUN22], [LR96], [BSE+23], but not compared with ensembled models [Eld03], [YZZ+13]. For ensembled models improved bootstrapping method proposed and compared with standard one.

4. Experimental research

In this section will present experimental research to uncover generalization properties of convolutional neural networks. All objectives of this work can be grouped into two categories: objectives that are related to convolutional neural network architecture topic and objectives that are related to data augmentation topic. In subsection 3.1 will be shortly presented datasets (training and testing) that has been used for the research. In subsection 3.2 comparing impact for generalization of two modern convolutional neural network architectures with different number of layers, creating ensembled model and comparing ability to generalize with plain architectures of selected convolutional neural networks. In subsection 3.3 covered the impact of data augmentation for generalization on different image levels. Accuracy metric was picked for generalization results comparison, an idempotent operation that simply divides accrued classified images and total count of samples, in all tables results are displayed in percents. To make assumption where generalization is better, calculated difference between the obtained accuracies.

4.1 Datasets used for research

In this work for training convolutional neural networks, it was used CIFAR-10, CIFAR-100 training datasets, because these two datasets are widely used for easy image classification task/benchmark in research community. The generalization of convolutional neural models refers to their ability to adapt to the new, previously unseen data that come from the same distribution as that used when the model was learned, for this purpose to test convolutional neural network ability to generalize it were used ImageNet and PASCAL VOC datasets.

CIFAR-10 and CIFAR-100, these datasets are labeled subsets of the 80 million tiny images dataset, they were collected by Alex Krizhevsky, Vinod Nair and Geoffrey Hinton [Kri09].

CIFAR-100 dataset is just like CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 super classes. List of classes in the CIFAR-100 in Table 1.

Table 1. CIFAR-100 dataset groups and classes.

Superclass	**Classes**
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea

Superclass	Classes
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Each image comes with a fine label, the class to which it belongs and a coarse label, the superclass to which it belongs.

The ImageNet dataset contains more than 14 million annotated images according to the WordNet hierarchy. Since 2010 the dataset has been used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a benchmark in image classification and object detection. The publicly released dataset contains a set of manually annotated training images (Fig. 17).

Figure 17. ImageNet dataset [STINDS23].

There are various subsets of the ImageNet dataset used in various contexts, but for this research, one of the most highly used subsets of ImageNet was used - "*ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012-2017 image classification and localization dataset*" [Ima17]. This dataset is also referred to in the research literature as ImageNet-1K or ILSVRC2017, reflecting the original ILSVRC challenge that involved 1,000 classes. ImageNet-1K contains more than 1 million training images, 50 thousand validation images and 100,000 test images. ImageNet is still one of the major datasets on which models are evaluated for their generalization in computer vision capabilities as the field moves towards self-supervised algorithms.

The PASCAL Visual Object Classes (VOC) 2012 dataset contains 20 object categories including: vehicles, household, animals, airplanes, bicycles, boats, cars, motorbikes, trains, bottles, chairs, dining tables, potted plants, sofas, TV/monitors, birds, cats, cows, dogs, horses, sheep, and people. Each image in this dataset has pixel-level segmentation annotations, bounding box annotations, and object class annotations. This dataset has been widely used as a benchmark for object detection, semantic segmentation, and classification tasks. The PASCAL VOC dataset is split into three subsets: 1464 images for training, 1449 images for validation and a private testing set (Fig. 18). Need to mention, that in this dataset all images contain at least one instance of each object category.

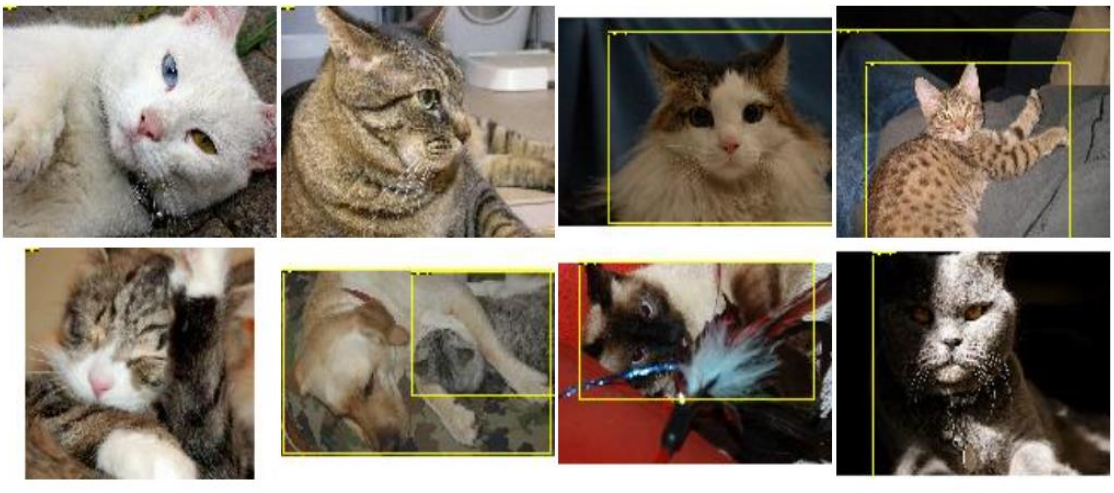

Figure 18. Pascal VOC 2012 dataset [PSVOC23].

4.2 Impact of CNN architecture on generalization

This section describes performed experimental researches, using 2 modern convolutional neural network architectures – ResNet and DenseNet. The first research is to compare the impact on generalization plain convolutional neural network architectures, the downsides of each architecture and practical implementation. The second research is to create ensembled network models from previously mentioned networks and compare the impact on generalization between ensembled and plain networks. In the process of creation ensembled network models to try to improve generalization by providing a new implementation to one of ensemble techniques.

Residual Network (ResNet) is a Convolutional Neural Network (CNN) architecture that overcame the vanishing gradient problem, making it possible to construct networks with up to thousands of convolutional layers, which outperform shallower networks (Fig. 19).

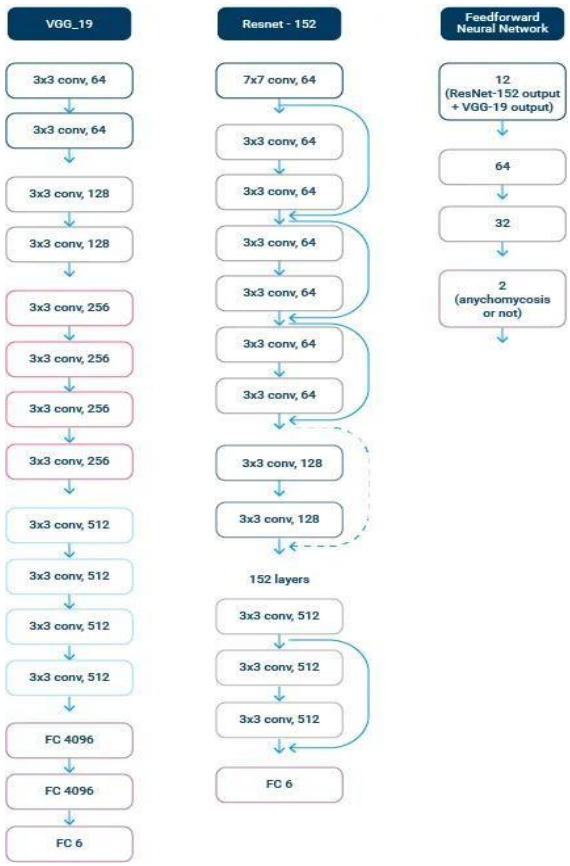

Figure 19. Residual learning: a building block [Kri09].

A vanishing gradient occurs during back-propagation. When the neural network training algorithm tries to find weights that bring the loss function to a minimal value, if there are too many layers, the gradient becomes very small until it disappears, and optimization cannot continue, but ResNet solved this problem using identity shortcut connections, it operates in two stages: firstly, it creates multiple layers that are initially not used, and skips them, reusing activation functions from previous layers, secondly, at a second stage, the network re-trains again, and the "residual" convolutional layers are expanded. This makes it possible to explore additional parts of the feature space which would have been missed in a shallow convolutional network architecture. Two stages are represented in Figure 19.

ResNet can contain many convolutional layers, commonly between 18-152, but supporting up to thousands of layers.

Each image comes with a fine label, the class to which it belongs and a coarse label, the superclass to which it belongs.

Residual blocks are the essential building blocks of ResNet networks. To make very deep convolutional structures possible, ResNet adds intermediate inputs to the output of a group of convolution blocks. This is also called skip connections, identity mapping, and "residual connections. The objective of skip connections is to allow smoother gradient flow, and ensure important features carry until the final layers. They do not add computational load to the network. The following diagram (Fig. 20) illustrates a residual block.

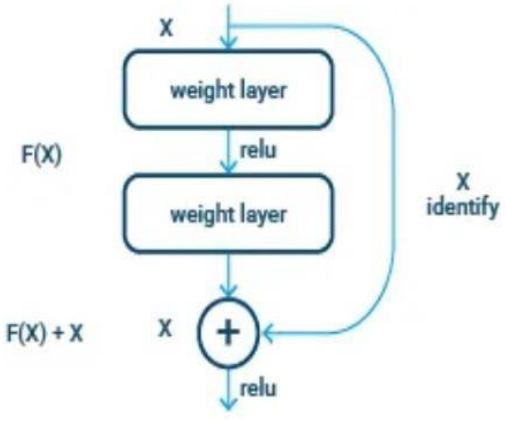

Figure 20. Residual learning: a building block [Kri09].

ResNet can contain many convolutional layers, commonly between 18-152, but supporting up to thousands of layers. In diagram (Fig. 20) X is the input to the ResNet block, the output from the previous layers and F(X) is a small neural network with several convolution blocks.

In ResNet, identity mapping is proposed to promote gradient propagation. Element-wise addition is used (Fig. 21). It can be viewed as algorithms with a state passed from one ResNet module to another one.

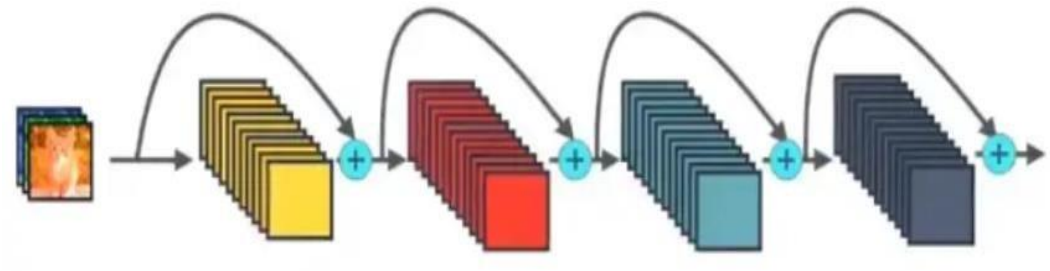

Figure 21. Element-wise addition [Tsa18].

In research titled "Deep Residual Learning for Image Recognition" by Kaiming He [HZ15], ResNet architecture is mainly explained for ImageNet dataset, since CIFAR datasets input images are 32x32 instead of 224x224 pixels, the upsampling was done.

Upsampling is a tool in post-production software to increase resolution. For upsampling it was used UpSampling2D class from Keras library with nearest interpolation value. The upsampling method accepts 2 integers, the upsampling factors for rows and columns.

Keras Applications include the following ResNet implementations and provide ResNet-V1 and ResNet-V2 with 50, 101 or 152 layers [KERESNET23]. The primary difference between ResNet-V2 and the original version 1 is that version 2 uses batch normalization before each weight layer. In this research version 2 was used.

Firstly, research has been done to get generalization results for convolutional neural network architectures with different number of layers and compare impact of different CNN architectures on generalization (all tests are run 20 times, Δ results are written).

Accuracy results for different testing datasets with plain ResNet network architecture, with different count of layers, were achieved and described in Table 2:

Table 2. Accuracy results with different ResNet architectures for different testing datasets.

Network	Training dataset	Accuracy with testing CIFAR-10 dataset	Accuracy with testing ImageNet-1k dataset	Accuracy with testing PASCAL dataset
ResNet with 50 layers	CIFAR-10	84.54	60.41	57.03
ResNet with 50 layers	CIFAR-100	77.23	64.28	59.75
ResNet with 101 layers	CIFAR-10	86.39	62.02	58.61
ResNet with 101 layers	CIFAR-100	79.82	65.89	60.07
ResNet with 152 layers	CIFAR-10	88.74	62.96	59.09
ResNet with 152 layers	CIFAR-100	80.39	66.74	61.06

As mentioned before, accuracy was picked as a main metric to make assumptions about the impact on generalization that provides network architecture. From obtained accuracies (Table 2) differences were calculated. For ImageNet-1k on CIFAR-10 training dataset the difference was 2.55 percent, on CIFAR-100 training dataset difference was 2.46 percent (compared models with different layers results). For Pascal dataset on CIFAR-10 training dataset the difference was 2.06 percent, on CIFAR-100 training dataset it was – 1.31 percent. The biggest results were expected from CIFAR-10 test dataset, because testing on same distribution dataset, for CIFAR-10 training dataset result was – 4.2 percent, for CIFAR-100 training dataset the difference was 3.16 percent. When comparing three ResNet networks with different number of layers we can notice a small increase on generalization and accuracy, because of increasing the depth increases the capacity of the model, but need to mention that increasing network depth does not work by simply stacking layers together. The downside of more layers - high computational requirements, increase the time it takes to train the network, the amount of training time can increase to the point that it is impossible to adequately train the convolutional neural network.

In this research for comparing with ResNet architecture were picked DenseNet network architecture [KERDANS23], because of feature use efficiency, fewer parameters that are described in Densely connected convolutional networks paper by Gao Huang [HLM+17].

Accuracy results for plain DenseNet network architecture with different count of layers were achieved and described in Table 3 and compared results with obtained ResNet results:

Table 3. Accuracy results with different DenseNet architectures for different testing datasets.

Network	**Training dataset**	**Accuracy with testing CIFAR-10 dataset**	**Accuracy with testing ImageNet-1k dataset**	**Accuracy with testing PASCAL dataset**
DenseNet with 121 layers	CIFAR-10	86.73	62.57	59.51
DenseNet with 121 layers	CIFAR-100	78.89	66.13	61.21
DenseNet with 169 layers	CIFAR-10	87.39	63.48	60.09
DenseNet with 169 layers	CIFAR-100	80.48	67.12	62.97
DenseNet with 201 layers	CIFAR-10	89.24	64.09	60.46
DenseNet with 201 layers	CIFAR-100	81.99	68.77	63.41

From obtained accuracies (Table 3) differences were calculated. For ImageNet-1k on CIFAR-10 training dataset the difference was 1.52 percent, on CIFAR-100 training dataset difference was 2.64 percent (compared models with different layers results). For Pascal dataset on CIFAR-10 training dataset the difference was 0.95 percent, on CIFAR-100 training dataset it was 3.9 percent. Results for CIFAR-10 test dataset from CIFAR-10 training dataset was 2.51 percent, for CIFAR-100 training dataset the difference was 3.1 percent. From differences we can notice that on smaller training dataset ResNet showed better results on generalization, but it is because accuracy from the starting point was lower compared with DenseNet, but when we are talking about bigger training dataset – DenseNet provides better results. When comparing ResNet and DenseNet plain convolutional neural networks according to accuracy, DenseNet network architecture is beating in RestNet architecture according by research data (Table 2 and Table 3). Different count of layers is tested also in Chaoning Zhang work [Zha21], very similar findings are made by author, that DenseNet architecture have more diversified features and tends to have richer patterns. Need to mention despite their competitive performance and overwhelming popularity, inherent drawbacks exist for both. For ResNet, the identity shortcut

that stabilizes training also limits its representation capacity, while DenseNet has a higher capacity with multi-layer feature concatenation, however, due to requiring high GPU memory it becomes very costly.

Every convolutional neural network model has its own weaknesses, so in this research convolutional neural network ensemble was also used to check impact on generalization and compare with plain convolutional neural networks results obtained in previous research. Convolutional neural network ensemble is a learning paradigm where many convolutional neural networks are jointly used to solve a problem (Fig. 22).

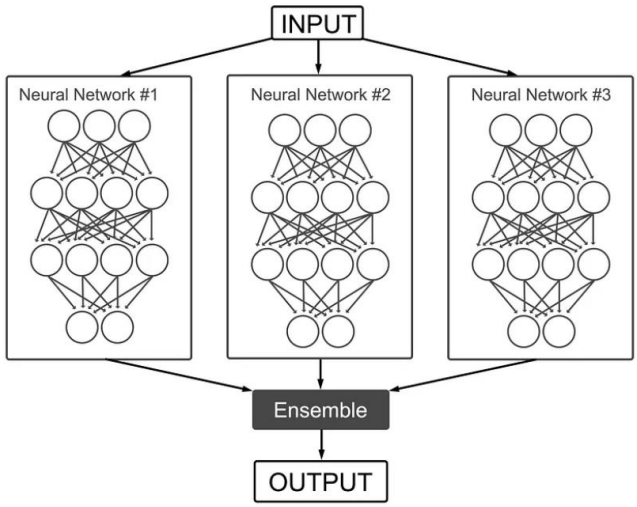

Figure 22. Ensemble of networks [ES23]

An ensemble model is one in which several other models, trained for the same task, are aggregated to generate a single output (Fig. 22), where all convolutional neural networks expect input images normalized in the same way. To implement ensembling Keras functional API library was used, and aggregation technique called – bootstrap.

The name bootstrap aggregating, also known as “bagging”, summarizes the key elements of this strategy. In the bagging algorithm, the first step involves creating multiple models. These models are generated using the same algorithm with random sub-samples of the dataset which are drawn from the original dataset randomly with bootstrap sampling method.

In bootstrap sampling, some original examples appear more than once, and some original examples are not present in the sample. To create a sub-dataset with m elements, you should select a random element from the original dataset m times, and if the goal is generating n dataset, it should be followed this step n times.

Accuracy results for ensembled ResNet and DenseNet models were achieved and described in Table 4 and compared below with single models with different number of layers:

Table 4. Accuracy results for ensembled ResNet and DenseNet models for different testing datasets.

Network	**Training dataset**	**Accuracy with testing CIFAR-10 dataset**	**Accuracy with testing ImageNet-1k dataset**	**Accuracy with testing PASCAL dataset**
Ensembled ResNet model	CIFAR-10	89.49	63.53	60.44
Ensembled ResNet model	CIFAR-100	82.06	67.02	61.79
Ensembled DenseNet model	CIFAR-10	90.73	65.03	60.72
Ensembled DenseNet model	CIFAR-100	83.11	69.04	63.88

From obtained accuracies (Table 4) differences were calculated (data also taken from Table 2 and Table 3). The ensembled ResNet model provided the following results: for ImageNet-1k test dataset on CIFAR-10 training dataset the difference was 0.57 percent, compared with ResNet model with 152 layers and 3.12 percent compared with ResNet model with 50 layers. On CIFAR-100 training dataset difference was 0.28 percent compared with ResNet model with 152 layers and 2.74 percent compared with ResNet model with 50 layers. For Pascal test dataset on CIFAR-10 training dataset the difference was 1.35 percent, compared with ResNet model with 152 layers and 3.41 percent compared with ResNet model with 50 layers. On CIFAR-100 training dataset difference was 0.78 percent compared with ResNet model with 152 layers and 2.04 percent compared with ResNet model with 50 layers. For test datasets that are not from CIFAR dataset distribution we can see, that created ensembled ResNet model provides better generalization compared to single models.

The ensembled DenseNet model provided the following results compared with DenseNet with 201 layers (that showed best results): for ImageNet-1k test dataset on CIFAR-10 training dataset the difference was 0.94 percent, on CIFAR-100 training dataset difference was 0.27 percent. For Pascal test dataset on CIFAR-10 training dataset the difference was 0.47 percent, on CIFAR-100 training dataset difference was 0.47 percent. From the results we can imply that ensembling DenseNet model provides slightly better results on generalization compared with DenseNet with 121, 169, 201 layers (Table 3), but in field such as healthcare, even the smallest amount of improvement in the generalization can be truly valuable. One of the downsides of ensemble is computing time, because the need to aggregate the output into single one.

Bootstrapping provides random sub-samples of the dataset which are drawn from the original dataset randomly with bootstrap sampling method, but it does not guarantee that models that are learning from that training data will generalize better compared with one more time bootstrapped dataset. Introduced data shuffling method that randomly switch images between split datasets, trying to increase diversity of it. Method accepts how many times to shuffle dataset and saving datasets and results in NoSQL database (Redis). Experiment was done with 20 shuffle times, follow results are described in Table 5:

Table 5. Accuracy results for ensembled ResNet and DenseNet architectures for different testing datasets with multiple data shuffle.

Network	**Training dataset**	**Accuracy with testing CIFAR-10 dataset**	**Accuracy with testing ImageNet-1k dataset**	**Accuracy with testing PASCAL dataset**
Ensembled ResNet model	CIFAR-10	90.56	64.21	61.04
Ensembled ResNet model	CIFAR-100	82.95	68.16	62.25
Ensembled DenseNet model	CIFAR-10	91.68	65.93	61.47
Ensembled DenseNet model	CIFAR-100	83.92	69.59	64.71

From Table 5 data we can imply that shuffling the datasets can provide better results compared to bootstrapping only once (Table 4), the highest difference is 1.14 percent, but need to mention the downside of it, is very long computing time and resources (memory) consumption, because needs to store data in database.

4.3 Impact of data augmentation on generalization

This section describes the impact of data augmentation for generalization properties of convolutional neural networks on different levels of images.

Data Augmentation is one of the best techniques for reducing overfitting to increase the size of the training dataset. As previously mentioned CIFAR-10 and CIFAR-100 are small image datasets, for experiment data augmentation was applied to training data. TensorFlow Core is providing data augmentation technique to increase the diversity of training set by applying random, but realistic, transformations, such as image rotation (Fig. 23).

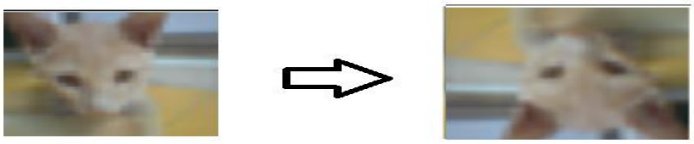

Figure 23. Augmented image, image rotation used.

To gray is an augmentation that converts the input color image to grayscale. For the grayscale image, each pixel contains only information about the amount of light within it. In other words, it carries only the intensity information. That is why grayscale images are monochrome and composed of shades of gray, from black at the pixel with the weakest intensity to white at the strongest (Fig. 24).

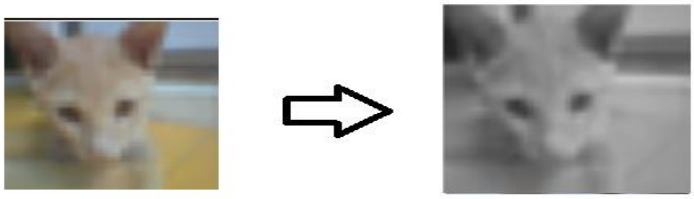

Figure 24. Augmented image, to gray used.

As for the exact algorithm behind the “to gray augmentation”, it is straightforward. Any color image is composed of three channels (red, green, and blue - RGB), whereas each channel contains information about the intensity of a particular color in a specific pixel. The idea is to remove all the color information leaving only the luminance values. With that done, your image will become monochrome composed of shades of gray. To gray is a valuable augmentation that can make the model less focused on color as a signal, and in some cases, it might be an intelligent decision. Jinyeong Wang in his paper „*Data Augmentation Methods Applying Grayscale Images for Convolutional Neural Networks in Machine Vision*“ [Wan21] noticing that if you are trying to detect an object that is always of the same color, to gray is not the best fit for you, because it bring very low generalization compared with time cost.

To increase the number of images and diversity of it, saturation was applied to training dataset (Fig. 25).

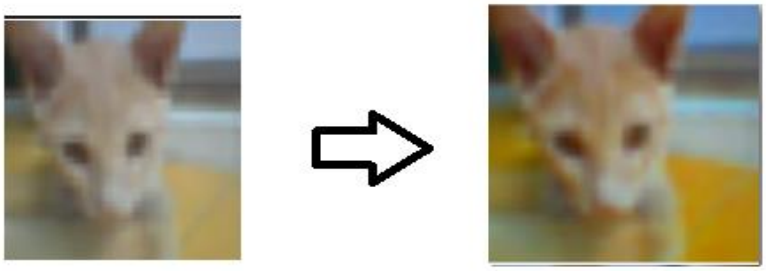

Figure 25. Augmented image, saturation applied.

This is a convenient method that converts RGB images to float representation, converts them to HSV, adds an offset to the saturation channel, converts back to RGB and then back to the original data type. If several adjustments are made, it is advisable to minimize the number of redundant conversions.

This is a convenient method that converts RGB images to float representation, converts them to HSV, adds an offset to the saturation channel, converts back to RGB and then back to the original data type. If several adjustments are made, it is advisable to minimize the number of redundant conversions.

Another data augmentation step was adjusting brightness of images because it is a common image augmentation method widely used in image processing. TensorFlow accepts images to adjust (RGB format) and delta, a scalar, amount to add to the pixel values (Fig. 26).

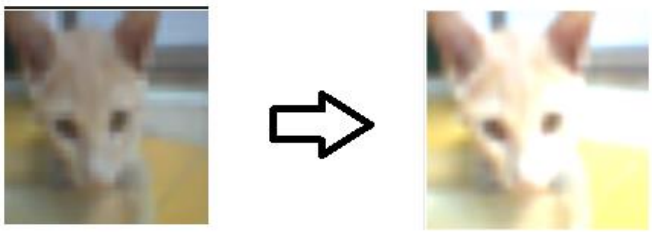

Figure 26. Augmented image, brightness applied.

This is a convenient method that converts RGB images to float representation, adjusts their brightness, and then converts them back to the original data type. If several adjustments are made, it is advisable to minimize the number of redundant conversions.

For rotating images by 90 degrees the rot90 method was used (Fig. 27). Method accepts a scalar integer tensor. The number of times the image is rotated by 90 degrees.

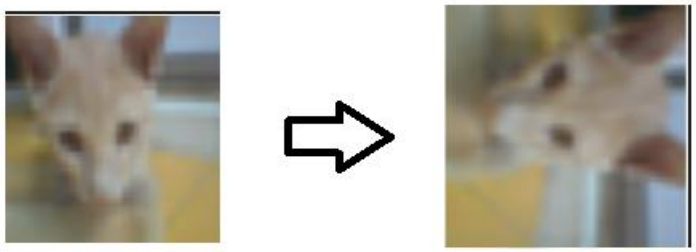

Figure 27. Augmented image, rotation applied.

These manipulations are done for CIFAR-10 and CIFAR-100 training datasets, tests were run 20 times on ImageNet-1k and PASCAL VOC testing datasets and Δ (delta) of results were calculated.

After these manipulations successfully managed to increase accuracy in all ResNet networks with different numbers of layers and in ensembled model. Addition of new data artificially derived from existing training data significantly improved accuracy, so generalization

is improved.:

Table 6. Accuracy results with different ResNet architectures where data augmentation was applied for different testing datasets.

Network	**Training dataset**	**Accuracy with testing CIFAR-10 dataset**	**Accuracy with testing ImageNet-1k dataset**	**Accuracy with testing PASCAL dataset**
ResNet with 50 layers	CIFAR-10	87.32	62.76	60.41
ResNet with 50 layers	CIFAR-100	79.21	67.06	61.89
ResNet with 101 layers	CIFAR-10	87.94	64.63	61.17
ResNet with 101 layers	CIFAR-100	80.94	68.51	62.66
ResNet with 152 layers	CIFAR-10	89.95	64.93	61.74
ResNet with 152 layers	CIFAR-100	83.54	69.02	62.95
Ensembled ResNet model	CIFAR-10	91.06	65.67	62.02
Ensembled ResNet model	CIFAR-100	84.28	69.76	63.51

From Table 6 all ResNet models showed better results with data augmentation (this table does not include ensembled model where multiple bootstrapping cycles were applied). From the beginning the DenseNet network architecture showed a higher score compared with ResNet architecture, so greater results are expected there. Data provided in Table 7 for architecture with data augmentation:

Table 7. Accuracy results with different DenseNet architectures where data augmentation was applied for different testing datasets.

Network	**Training dataset**	**Accuracy with testing CIFAR-10 dataset**	**Accuracy with testing ImageNet-1k dataset**	**Accuracy with testing PASCAL dataset**
DenseNet with 121 layers	CIFAR-10	88.24	63.32	60.96
DenseNet with	CIFAR-100	81.37	67.74	62.07

121 layers				
Network	**Training dataset**	**Accuracy with testing CIFAR-10 dataset**	**Accuracy with testing ImageNet-1k dataset**	**Accuracy with testing PASCAL dataset**
DenseNet with 169 layers	CIFAR-10	90.03	64.16	61.02
DenseNet with 169 layers	CIFAR-100	83.82	68.15	63.17
DenseNet with 201 layers	CIFAR-10	91.42	65.33	61.76
DenseNet with 201 layers	CIFAR-100	85.76	69.12	63.83
Ensembled DenseNet model	CIFAR-10	91.67	66.01	62.14
Ensembled DenseNet model	CIFAR-100	87.08	70.09	64.62

From data from Table 6 and Table 7 we can see that DenseNet network architecture generalizes better compared with ResNet network architecture when comparing accuracies differences, it is because network diminishes the vanishing gradient problem, and it requires fewer parameters to train the model. Dynamic featurepropagation takes care of the seamless flow of information. Results can be improved by fine- tuning the model. To try adding or removing more dense blocks and layers, finding the frequencyof data in each class.

Standard data augmentations are mostly performed at the image level, which yields all-round performance improvement in generalization. Usually, image level augmentation preserves semantics globally, following humans' cognitive intuition. However, how to perform data augmentation at a non-image level, in other words, at patch level or piece level, is rarely studied [HFL+22]. For a specific augmentation, we may perform this augmentation on these pieces individually and combine transformed pieces back to a single image. Such a strategy should increase the diversity in both local regions level as well as at the holistic image level and may also encourage neural networks to share the same cognitive ability of recognizing objects from partial information like humans can.

In this research part, one image was cut to x parts, where x is number that divide by 2 without remainder (x mod 2 = 0, where x > 0) and width or length is > 0, was investigated what impact it has for generalization, because hypothesis was that such a strategy should increase the

diversity in both local regions level as well as at the holistic image level and may also encourage convolutional neural networks to share the same cognitive ability of recognizing objects from partial information like humans can.

Implemented method (there is no open-source code that would do the same) cuts one image into x equal pieces, either in the height or the width dimension. The same data augmentations are performed independently with some rules within each piece. Augmented pieces are then concatenated together to form one single augmented image. Figure 28 shows horizontally cut of one image into 2 equal pieces.

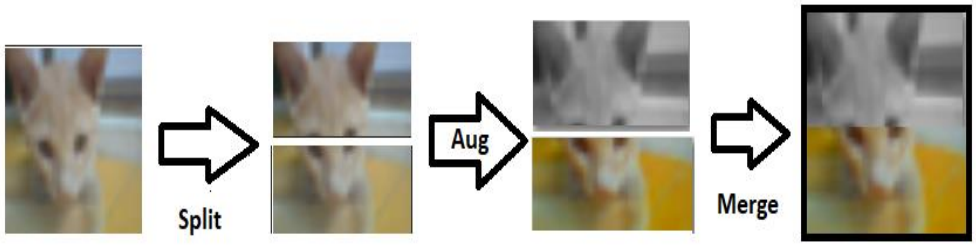

Figure 28. Image augmentation on non-image level.

Image can be cut horizontally or vertically (parameter providing as a Enum to created method). For each piece augmentation is applied, there is a rule – if one cut gets horizontal flip, rotation, resizing – another piece gets it also and second augmentation is applied once again. This is done to verify that the image will not be corrupted and will have completeness, because data is generally considered high quality if it is well suited to serve its specific purpose. This method with data augmentation was applied to the whole CIFAR-10 and CIFAR-100 training datasets, to keep balanced dataset, because unbalanced dataset, can introduce bias, an anomaly where the model inherits prejudices in the training data.

The data provided in Table 8 are results of accuracy where image is cut into 2 pieces, augmentation applied, and merged into one image, both horizontal and vertical cuts were done, and best results are taken:

Table 8. Accuracy results with different ResNet architectures for different testing datasets.

Network	Training dataset	Accuracy with testing CIFAR-10 dataset	Accuracy with testing ImageNet-1k dataset	Accuracy with testing PASCAL dataset
ResNet with 50 layers	CIFAR-10	87.74	63.22	61,06
ResNet with 50 layers	CIFAR-100	79.95	68.11	62.68
ResNet with 101 layers	CIFAR-10	88.49	65.07	62.44
ResNet with 101 layers	CIFAR-100	81.39	69.26	63.57
ResNet with 152 layers	CIFAR-10	90.47	65.99	62.82
ResNet with 152 layers	CIFAR-100	83.94	70.68	64.21
Ensembled ResNet model	CIFAR-10	91.66	66.24	63.19
Ensembled ResNet model	CIFAR-100	84.82	71.17	64.73
DenseNet with 121 layers	CIFAR-10	88.95	65.58	63.02
DenseNet with 121 layers	CIFAR-100	81.89	70.42	64.27
DenseNet with 169 layers	CIFAR-10	91.06	66.04	63.76
DenseNet with 169 layers	CIFAR-100	84.25	70.79	64.98
DenseNet with 201 layers	CIFAR-10	92.03	67.15	64.21
DenseNet with 201 layers	CIFAR-100	86.36	71.16	65.74
Ensembled DenseNet model	CIFAR-10	92.14	67.63	64.77
Ensembled DenseNet model	CIFAR-100	88.27	71.59	66.06

From Table 8 data we can see that cutting image into several pieces, applying augmentation, and combine transformed pieces back to a single image can lead to good results comparing with no-augment training sets and where image-level augmentation was applied (from Table 2 to Table 7), because of increased diversity of in both local regions level as well as at the

holistic image level.

From previous experiments we already know that ensembled DenseNet model showed the best results on generalization, this model was chosen to check what impact cutting into n pieces have to generalization. Following experiment was done, images were cut into 2, 4, 8 pieces, results are displayed in Table 9:

Table 9. Accuracy results with different cuts count on image for different testing datasets.

Network	**Accuracy with testing CIFAR-10 dataset**	**Accuracy with testing ImageNet-1k dataset**	**Accuracy with testing PASCAL dataset**
CIFAR-10 and image cut into 2 pieces	92.14	67.63	64.77
CIFAR-100 and image cut into 2 pieces	88.27	71.59	60.06
CIFAR-10 and image cut into 4 pieces	92.98	68.16	65.33
CIFAR-100 and image cut into 4 pieces	89.27	71.97	63.79
CIFAR-10 and image cut into 8 pieces	91.83	66.51	63.79
CIFAR-100 and image cut into 8 pieces	87.56	71.04	58.96

Data provided from Table 9 shows that more cuts are not necessary to bring more generalization, otherwise cut for 8 pieces performs poorly compared with two other methods, mostly it depends on the content and the nature of the image. In general – cutting method brings better generalization, because of bringing more diversity to dataset, because applied not only in image-level, but in piece-level too, but needs to mention biggest downside – computing time, more pieces and more augmentation need to apply, bigger the computing time and more computer resources are required.

Results and conclusions

In this master's thesis, the following was performed:

1) Research has been done on how convolutional neural network architecture can make impact of generalization, performed, and compared two modern architectures and their impact on training and generalization.

2) Research has been done on how convolutional neural networks ensembled model perform on generalization compared with plain architectures and how that impact training and generalization.

3) Applied different data augmentation techniques on image level to retrieve better generalization of convolution neural networks.

4) Research has been done on the generalization impact of non-image level augmentation, compared with augmentation on image level results.

As a result of research in this master's thesis, the following conclusions were obtained:

1) Convolutional neural network architecture has impact to generalization, all architectures are trying to improve their predecessors. In this research the resources and learning time were not in the count, when comparing generalization through accuracy perspective, DenseNet performing in overall better than ResNet (showing up to 2.59 percents better generalization, results with Pascal test dataset). Need to mention that DenseNet network from the beginning had better accuracy score, so it might lead to better generalization. For future improvement – more metrics can be investigated.

2) Convolutional neural networks ensemble model outperforms on generalization single model network architectures. In this research 2 ensemble models were made, one is from ResNet models, where they are jointly to solve a problem, another - from DenseNet models. Both showed better results on generalization, because individually each convolutional neural network has its own weaknesses, but together when they aggregated to generate single output – they are showing slightly better generalization. Introduced shuffling dataset method for ensemble bootstrapping technique that increased generalization by 1.14 percent. The downsides –

computing time, because need to wait for all responses to aggregate into single one and occupied memory, because of storing data of each bootstrapped dataset.

3) The data augmentation on image level has impact on generalization, for small datasets require to apply data augmentation to make the data rich and sufficient and thus makes the model perform better, it helps in recognizing samples the model has never seen before.

4) The data augmentation on non-image level has impact on generalization. In this research the experiment was done when the image was cut into *n* pieces and for each piece – data augmentation was applied, after that – it was concatenated into single image once again. Several rules were applied to keep image not corrupted, for example: cannot apply rotation to single piece. Results show that this kind of manipulation on non-image level provides increase of the diversity in both local regions level and as well as at the holistic image level, that brings better generalization compared to traditional data augmentation on image level. There is a hook – cutting a single image into a lot of pieces does not always provide a good generalization, it depends on content and nature of the image (size and what percent of the image contains the object), because in another case – scalability of this method would be big. The downside of this cutting technique is computing time, because for each piece need to apply augmentation and then bring back for concatenation. This part can be improved because the implementation is written by a researcher that contains source code.

References

[TK99] Kavzoglu, Taskin. "Determining optimum structure for artificial neural networks." Proceedings of the 25th Annual Technical Conference and Exhibition of the Remote Sensing Society. Remote Sensing Society Nottingham UK Cardiff, UK, 1999.

[PP18] Potkin, Oleg, and Andrey Philippovich. "Static Gestures Classification Using Convolutional Neural Networks on the Example of the Russian Sign Language." AIST (Supplement). 2018.

[Hus06] Sazli, Murat H. "A brief review of feed-forward neural networks." Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering 50.01 (2006).

[Rad17] Radiuk, Pavlo M. "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets." Information Technology and Management Science 20.1 (2017): 20-24.

[Fuk80] Fukushima, Kunihiko. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." Biological cybernetics 36.4 (1980): 193-202.

[HLM+17] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[CSW+19] Cai, Shaofeng, et al. "Effective and efficient dropout for deep convolutional neural networks." (2019). (nėra nuotraukų)

[AGH+17] Aquino, N. Romero, et al. "The effect of data augmentation on the performance of convolutional neural networks." Braz. Soc. Comput. Intell (2017).

[LBB+98] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[STI+18] Santurkar, Shibani, et al. "How does batch normalization help optimization?" Advances in neural information processing systems 31 (2018).

[HZ15] He, Kaiming, et al. "Microsoft Research: Deep Residual Learning for Image Recognition." (2015).

[GG17] Gitman, Igor, and Boris Ginsburg. "Comparison of batch normalization and weight normalization algorithms for the large-scale image classification." (2017).

[TAL16] Targ, Sasha, Diogo Almeida, and Kevin Lyman. "Resnet in resnet: Generalizing residual architectures." (2016).

[NBM+17] Neyshabur, Behnam, et al. "Exploring generalization in deep learning." Advances in neural information processing systems 30 (2017).

[EA21] Ebrahimi, Mohammad Sadegh, and Hossein Karkeh Abadi. "Study of residual networks for image recognition." Intelligent Computing: Proceedings of the 2021 Computing Conference, Volume 2. Springer International Publishing, 2021.

[TY20] Takagi, Shiro, Yuki Yoshida, and Masato Okada. "The Effect of Batch Normalization in the Symmetric Phase." Artificial Neural Networks and Machine Learning–ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part II 29. Springer International Publishing, 2020.

[LS18] Luo, Ping, et al. "Towards understanding regularization in batch normalization." (2018).

[LZZ21] Li, Yang, Hong Zhang, and Yu Zhang. "Rethinking training from scratch for object detection.". 2021.

[Hay8] Haykin, Simon. Neural networks: a comprehensive foundation. Prentice Hall PTR, 1998.

[Kri09] Krizhevsky, Alex. "Learning multiple layers of features from tiny images." (2009): 7.

[LBD+89] LeCun, Yann, et al. "Backpropagation applied to handwritten zip code recognition." Neural computation 1.4 (1989): 541-551.

[HLM+17] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[Zha21] Zhang, Chaoning, et al. "Resnet or densenet? introducing dense shortcuts to resnet." Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2021.

[Wan21] Jinyeong Wang. Data Augmentation Methods Applying Grayscale Images for Convolutional Neural Networks in Machine Vision. 2021.

[Pra21] Pramoditha, R. "The Concept of Artificial Neurons (Perceptrons) in Neural Networks." (2021).

[STINDS23] ImageNet dataset. Interactive [accessed 2023-05-07].

URL:< https://cs.stanford.edu/people/karpathy/cnnembed/>

[PSVOC23] PASCAL VOC dataset. Interactive [accessed 2023-05-07].

URL:< http://host.robots.ox.ac.uk/pascal/VOC/voc2012/examples/index.html#bird>

[KERESNET23] Image. Interactive [accessed 2023-03-10].

URL:< https://keras.io/api/applications/resnet/>

[HFL+22] Han, Junlin, et al. "You only cut once: Boosting data augmentation with a single cut." International Conference on Machine Learning. PMLR, 2022.

[LZ16] Hongtao, Lu, and Zhang Qinchuan. "Applications of deep convolutional neural network in computer vision." Journal of Data Acquisition and Processing 31.1 (2016): 1-17.

[XCW+21] Xie, Tianshu, et al. "Cut-thumbnail: A novel data augmentation for convolutional neural network." Proceedings of the 29th ACM International Conference on Multimedia. 2021.

[IS15] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. pmlr, 2015.

[SKP18] Sewak, Mohit, Md Rezaul Karim, and Pradeep Pujari. Practical convolutional neural networks: implement advanced deep learning models using Python. Packt Publishing Ltd, 2018.

[Jia23] Jiang, Yuan. "Exploring The Efficiency of Resnet and Densenet in Gender Recognition." *Highlights in Science, Engineering and Technology* 38 (2023): 1033-1037.

[Ima17] ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Interactive [accessed 2023-05-17].

URL:< https://www.image-net.org/challenges/LSVRC/>

[LC18] Liu, Yin, and Vincent Chen. "On the Generalization Effects of DenseNet Model Structures." (2018).

[Kha21] Biswas, Angona, and Md Saiful Islam. "An efficient CNN model for automated digital handwritten digit classification." Journal of Information Systems Engineering and Business Intelligence 7.1 (2021): 42-55. (pakeisti reikia)

[RGY+23] Raileanu, Roberta, et al. "Automatic data augmentation for generalization in reinforcement learning." Advances in Neural Information Processing Systems 34 (2023): 5402-5415.

[Eld03] Elder IV, John F. "The generalization paradox of ensembles." Journal of Computational and Graphical Statistics 12.4 (2003): 853-864.

[SL20] Sediqi, Khwaja Monib, and Hyo Jong Lee. "Improved Image Semantic Segmentation Based on Cascade Data Augmentation." 2020 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, 2020.

[SM21] Saini, Dipen, and Rahul Malik. "Image Data Augmentation techniques for Deep

Learning-A Mirror Review." 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO). IEEE, 2021.

[LLL+21] Li, Pan, et al. "A simple feature augmentation for domain generalization." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021.

[AUN22] Adil, Mohammad, et al. "Effect of number of neurons and layers in an artificial neural network for generalized concrete mix design." Neural Computing and Applications (2022): 1-9.

[LR96] Littmann, Enno, and Helge Ritter. "Learning and generalization in cascade network architectures." Neural Computation 8.7 (1996): 1521-1539.

[BSE+23] Bonfanti, Andrea, et al. "On the Hyperparameters influencing a PINN's generalization beyond the training domain." (2023).

[YZZ+13] Yang, Jing, et al. "Effective neural network ensemble approach for improving generalization performance." IEEE transactions on neural networks and learning systems 24.6 (2013): 878-887.

[Guo11] GUO, Hongwei. A simple algorithm for fitting a Gaussian function [DSP tips and tricks]. IEEE Signal Processing Magazine, 2011, 28.5: 134-137.

[Tsa18] Tsang, Sik-Ho. Image. Interactive [accessed 2023-04-28] at “Review: DenseNet — Dense Convolutional Network (Image Classification)”. 2018.

URL:< https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803>

[CAG+15] Cogswell, Michael, et al. "Reducing overfitting in deep networks by decorrelating representations." arXiv preprint arXiv:1511.06068 (2015).