Exploring Convolutional Architecture Capabilities for Image Classification Tasks with Insufficient Amount of Data[*]

Andrii Matsevytyi[1,*,†] and prof. Vytautas Rudzionis[1,†]
[1] Vilnius University, Kaunas Faculty, Muitinės g. 8, Lithuania

Abstract
Nowadays Convolutional Neural Networks are used everywhere from facial recognition to malware detection and flat evaluation and are considered to bring significant changes to computer vision. They introduce solutions of such problems as insufficient and low-quality dataset. However, they tend to possess same problems as other Machine Learning and Deep Learning techniques. The paper considers and analyses the most commons methods for image classification, involving usage of feed-forward convolutional architecture. The object of the study is self- collected dataset, consisting of 7 classes, that provide of low-, middle- and high-level features. The subject of the study is to explore the capabilities of CNNs key architecture blocks and their combinations.

Keywords
Convolutional neural networks, image classification, kernel, pooling, accuracy metrics, optimization, low quality dataset, small dataset

1. Introduction

Classifying images is one of fundamental tasks in Machine Learning and Data Analysis. It plays a crucial role in our everyday life – from facial recognition[1] to flats filtering, evaluation and classification[2]. There are very many image recognition methods used by various researchers. Among them, can be mentioned linear discrimination methods, nearest neighbours approach, SVM and other approach. Using these methods different degrees of success were achieved for various tasks. The relative drawback of many of those methods is the necessity to use somehow derived features. Many feature extraction procedures have been proposed for use in a biometric system, including principal component analysis (PCA), independent component analysis (ICA), local binary patterns (LBP), the histogram method and others.

However, these methods require big amount of good quality data to achieve some competitive benchmark. And nowadays there are fields where it is very hard to collect such dataset. A good example is the task of recognizing the symbols, widely used by various youth subcultures. There are amateur periodicals containing images with those symbols but the amount of available images is restricted. Additionally, many of those images are of low quality since were done (captured or painted) by amateur authors using not the best techniques available. This means that it is hard to increase the available amount of data for training. But as good recognition as possible could be of big help for the people interested in these youth groups, in particular psychologists and anthropologists.

Convolutional Deep Neural Networks appear to be extremely powerful tool in image classification. In fact, they have revolutionized computer vision[3]. The choice of convolution and pooling in CNNs is motivated by the desire to endow the networks with invariance to irrelevant cues such as image translations, scalings, and other small deformations[4-5]. For the task of helping the people working in these areas, CNN seems to be one of the best approaches due to its high efficiency and robustness. At the same time, it is necessary to find the best way to solve the particular task of symbol recognition using low-quality data of limited amount.

The aim of this work is to conduct research on how different layers and their combinations influence the accuracy of the convolutional network in the context of elusive dataset and to address such issues as

[*] *IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania*
[1,*] Corresponding author
[†] These author contributed equally.

andrii.matsevytyi@knf.stud.vu.lt (A. Matsevytyi), vytautas.rudzionis@knf.vu.lt (V. Rudzionis)
0009-0005-1781-5040 (A. Matsevytyi); 0009-0007-5683-7546 (V. Rudzionis).

overfitting predisposition and usage of small dataset of insufficient quality. Here main layers of convolutional network, their hyperparameters and capabilities of extracting different and similar low-level, middle-level and high-level features in image classification tasks will be overviewed.

2. Related works

This research was inspired by Chiheb Chebbi's book "Mastering Machine Learning for Penetration Testing", particularly in chapter 4, where CNN is used to detect malware applications[7]. One of the first successful approaches in using CNN for image classification is LeNet-5, introduced in 1989, where a simple CNN was used for handwritten digits recognition[8]. This was a simple model, but it became a powerful tool and with best test error rate of less than 0.3% approached the human level[9]. However, while being good at digits recognition, that time CNN-like approaches brought pure capabilities in real-world scenarios[10]. Probably the first powerful example to overcome the issue is AlexNet, introduced by Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton in 2012 [11]. It achieved only 17% loss on top-5 error rate[2] and was a breakthrough in this field, despite in fact it looked similar to LeNet-5, having 9 times bigger input height and width as well as additional Convolutional and Dense layers. The next approach taken was using similar architecture but pushing model depth to 16-19 model layers, and as a result, VGG model appeared with result of 7.3% top-5 error [12].

Convolutional Neural Networks still face few weaknesses, in particular issues with transfer learning, limitations of interpretability and computational complexity. These drawbacks are not covered in the paper. This paper will try to address the issues, related to overfitting predisposition and usage of small dataset of insufficient quality.

2.1 Related theory

A convolutional neural network is a deep learning model for processing data that has a grid structure, for example, images. Appearance of CNNs was inspired by the organization of the visual cortex of the brain of animals and designed for automatic and adaptive learning and extraction of the spatial hierarchy of features present in the image, from low features level (i.e. angles, straight lines, horizontal lines) to more complex ones, high level patterns.[13]

This network usually consists of three types of layers (groups of neurons): convolution, pooling (pooling, subsampling) and fully connected layers, as a reduced fully connected neural network. The first two types of layers, convolution, and pooling, perform the function of feature extraction, while fully connected layers translate the extracted features into a final result, such as the probabilities of image belonging to classes in the case of a classification task.

Convolutional Layer

Convolutional layer is the cornerstone of all convolutional networks. This layer applies sliding of different kernel filters to capture different patterns and consists of combination of linear and non-linear operations – convolution operations and activation functions.

Convolution is a special type of linear operation, used for feature extraction, where a small array of numbers, called a kernel, is applied to the input data (which is also an input data).

[2] The "top-5 error" is the percentage of times that the target label does not appear among the 5 highest- probability predictions, and many methods cannot get below 25%

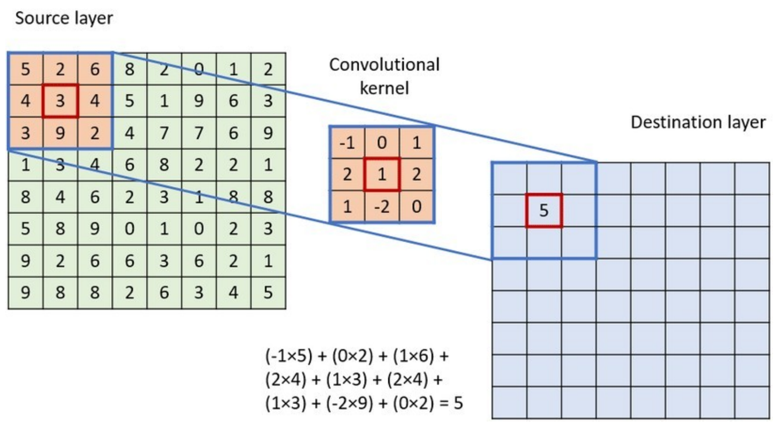


Figure 1: Schematic illustration of convolution computation [6]

Elemental product between each core element and the input element by the tensor is calculated at each location of the tensor and summed for obtaining the output value in the corresponding position of the output tensor, which is called a feature map. The procedure is repeated using several cores for formation of any number of feature maps that extract various tensors of input characteristics. There are three types of kernels based on the features, they can extract: low-level, middle level and high-level ones. Low-level (edge detection) kernels or are used to capture low-level features (edges, corners, simple textures). They help in extracting basic visual patterns and together form more advanced features. To move from lower to higher level kernel, few lower level kernels are combined. Mid-level (pattern detection) kernels focus on capturing more complex textures, patterns, and shapes. High-level (object detection or semantic feature) kernels detect whole structures and are learned during training in deep learning models. Also, based on the task specifics, sometimes custom kernels, designed for the particular task, are applied. Below there is an illustration of instances of different kernels.

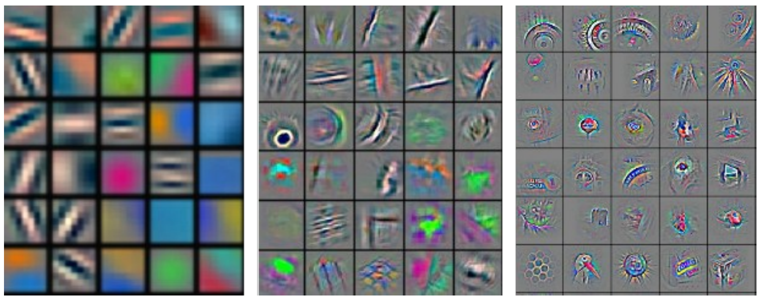


Figure 2: Low-level, mid-level and high-level kernels example (source - https://stats.stackexchange.com/questions/362988/in-cnn-do-we-have-learn-kernel-values-at-every-convolution-layer)

Convolutional layer usually takes such hyperparameters as number of kernels, kernel size, stride length and padding. Kernel amount means amount of different kernels used in convolution. It directly influences amount of features that can be extracted, allowing the network to learn more complex features but also increasing time and computational complexity of training. Kernel size specifies the dimensions, used in convolution process. Common dimensions are 2/2, 3/3, 5/5 and 7/7. Bigger kernel is able to capture more global features, removing the noise or small-scale features. Stride length means step that kernels take while sliding. Bigger stride means decreased complexity of computations, but also leads to loosing some small features. Padding helps in better revealing features on edges of picture by creating an additional row around the image.

Pooling Layer

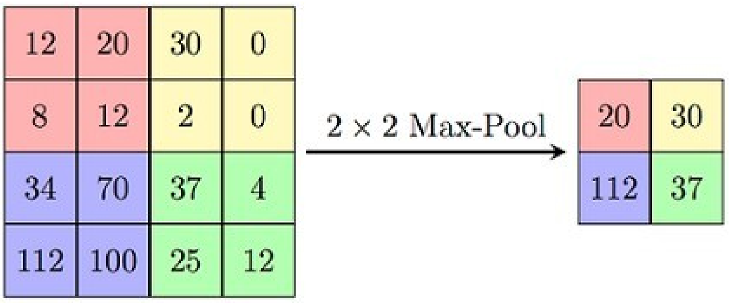


Figure 3: Short representation on how Max Pooling works (source - https://paperswithcode.com/method/max-pooling)

Pooling layer has a task of downsampling dimensions of image while preserving the necessary features. The output is called a Pooled Feature Map. Main reasons to apply such layer is to decrease overfitting and computational cost.

The most popular form of aggregation operation is Max Pooling, which extracts patches from the input feature maps and proceeds with the maximum value in each patch.

In practice, maximum aggregation is traditionally used from the core (filter) of size 2 × 2 with a step of 2. This helps reducing main dimensions in two times. Unlike height and width, depth feature size remains unchanged since pooling is done for each layer patch in depth separately.

Fully Connected Layer (Flatten and Dense Layers)

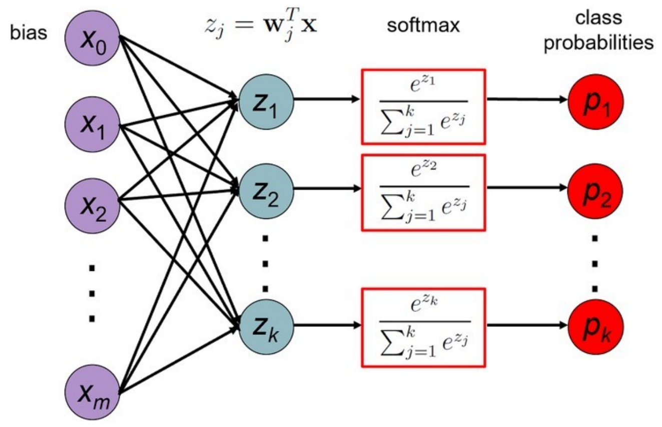

bias
$z_j = \mathbf{w}_j^T \mathbf{x}$
softmax
class probabilities
x_0
x_1
x_2
x_m
z_1
z_2
z_k
$\frac{e^{z_1}}{\sum_{j=1}^k e^{z_j}}$
$\frac{e^{z_2}}{\sum_{j=1}^k e^{z_j}}$
$\frac{e^{z_k}}{\sum_{j=1}^k e^{z_j}}$
p_1
p_2
p_k

Figure 4: Short representation on how the last Dense layer performs the classification
(source - https://towardsdatascience.com/deep-dive-into-softmax-regression-62deea103cb8)

The output feature maps of the last layer of convolution or pooling are usually flatten, i.e. turn into a one-dimensional array of numbers, and connected to one or more fully connected layers. Here each input is being associated with each output by the weights that are learned. After creating patterns of features, extracted by convolution layers and reduced by layers of aggregation (pooling), they are connected by a subset of fully connected layers with class probabilities as final network outputs.

Commonly used activation functions

Activation functions are used to choose the best option from the given features. In neural network each neuron in same layer has same activation function. As model is trained by computing gradient descent and backpropagating it through error signals for each neuron, and CNN consists of millions of neurons, to simplify the computational complexity, for hidden layers usually ReLU (Rectified Linear Unit) function is used. Also, due to its capabilities to handle values, dropping below 0, sometimes Leaky ReLU is used[14]. One of examples of such situation are networks with a prevailing number of negative inputs.

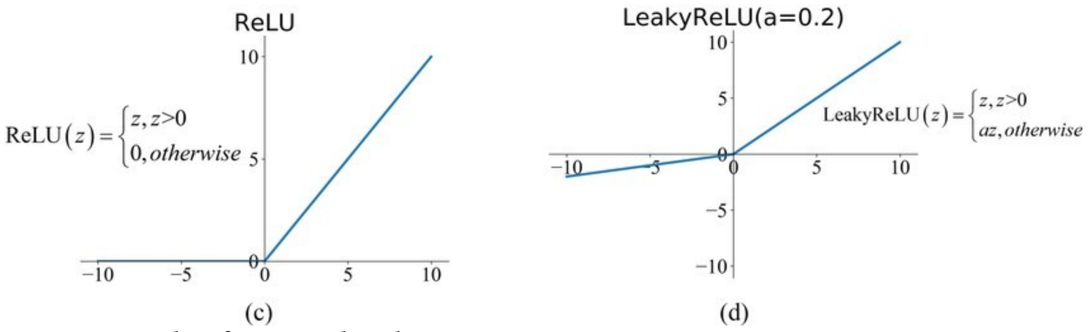


Figure 5: graphs of ReLU and Leaky ReLU
(source - https://www.researchgate.net/figure/a-The-functional-curve-of-the-activation-function-s-and-b-the-functional-curve-of_fig7_347565853)

Output layer is responsible for classifying signals, passed from hidden layers into classes. Depending on the task, either softmax or logistic activation function may be used. Softmax is used for multi-class classification, while logistic is its version for binary classification. Tanh (Hyperbolic Tangent) maps the output to the range (-1, 1), and this brings benefits in some situations. [15-16] It is symmetrical in comparison to softmax/logistic function and is claimed as being more balanced in binary classification tasks. Also, tanh 0 as the fastest point (representing highest gain), and for logistic 0 is the lowest point and it becomes a trap for anything going below.

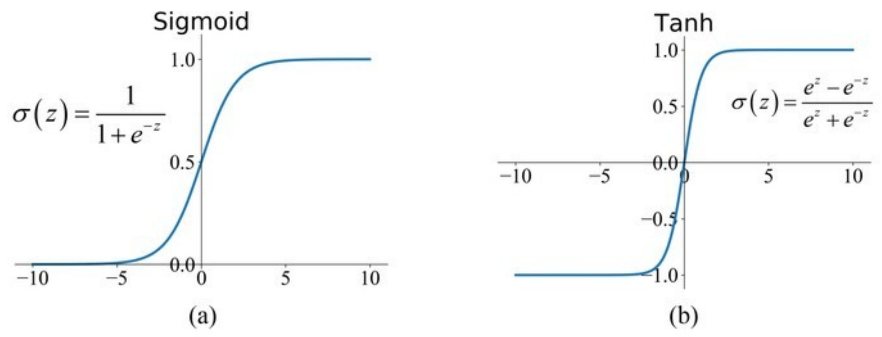


Figure 6: graphs of Sigmoid (softmax) and Tanh activation functions
(source - https://www.researchgate.net/figure/a-The-functional-curve-of-the-activation-function-s-and-b-the-functional-curve-of_fig7_347565853)

3. Deep learning model

In this study, different architectures of CNNs will be tested to figure out their capabilities for classifying different level features on a low-quality dataset with an insufficient amount of data. The choice of a Convolutional Neural Network (CNN) architecture is particularly suitable for this study due to its inherent ability to extract hierarchical features from images, which is crucial for classifying symbols and patterns in low-quality datasets. CNNs are designed to capture spatial relationships in data, making them robust to variations such as image translations, scaling, and deformations, which are common challenges in recognizing symbols captured by amateur authors using diverse techniques. Also, convolutional approach is chosen due to its ability of learn on comparingly small amounts of data.

The initial configuration of the model is inspired by the LeNet-5 architecture[7]. It is also influenced by AlexNet approach[11], due to its good performance in image classification tasks on large datasets. However, in this research the task provides dataset with small amount of classes and insufficient amount of data with not the best quality, therefore the original AlexNet may be not the best solution and higher-scale approaches such as VGG[12] is not taken into account. However, there will be modified the number of layers and their configurations to explore different model complexities and performance outcomes. All models consist of combinations of Conv2D and MaxPooling2D layers, followed by Flatten and Dense layers. For the Conv2D layer, a 3/3 kernel is used. To cover all possible amounts of default kernel variation, each time the amount of kernels will be doubled, starting from 32. For the MaxPooling2D layer, a 2/2 kernel will be used.

Firstly, different amount of pairs of Convolutional-MaxPooling layers will be tested. Since dataset is small, the best approach should be having two or three pairs of Conv2D+MaxPooling2D layers. Then, the best result is taken, analysis of its performance is done, and additional Convolutional Layer is added in different parts of network with further analysis. Eventually different amount of Dense layers is tried to better reveal how captured features were useful in this task.

4. Experimental setup

4.1 Dataset and data preprocessing

Due to the lack of existing datasets in this filed, for testing the capabilities of the neural network is used a self-collected balanced dataset, which consists of 700 open-source images, belonging to 7 classes: Horn gesture, Anarchy graffiti, Pentagram, Inverse Pentagram, Scull, Panks and Metalists. Below are examples of each dataset:

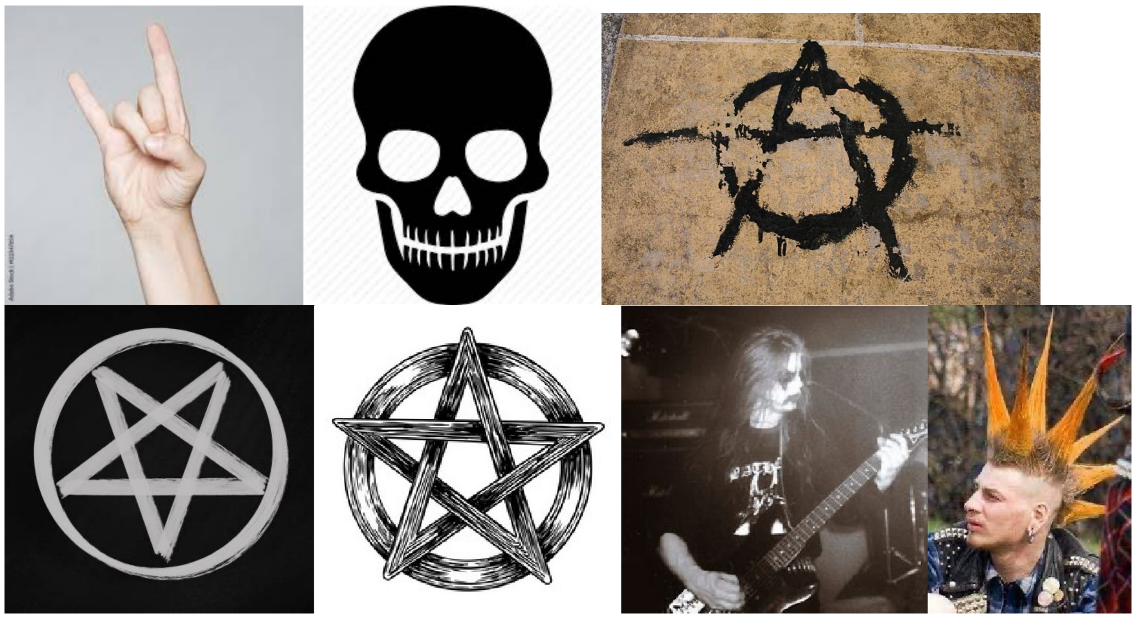


Figure 7: Example of the dataset that is used

Pentagram, Anarchy and Inverse Pentagram mainly require low-level features to be extracted. Horn gesture and Scull contain mostly middle-level features, while Panks and Metalists have high-level features prevailing. During data preprocessing, each photo is converted to shape (244, 244, 3) – to square 244/244 pixels with 3 color (RGB) channels, similarly to how it is done in AlexNet[11]. After that in order to decrease model overfitting, there is applied augmentation[3], a very powerful technique in reducing overfitting[17]. In particular, the next techniques were applied:

Table 1
Applied augmentation techniques

Augmentation type	Value 0-10%
Width shift	0-10%
Height shift	0-20%
Shear range Zoom	0-20%
range Flip	Horizontal Nearest
Fill mode	

4.2 Software and hardware stack that was used

For experiments in this work was used Kaggle Environment – a cloud-based Python environment with access to x2 Tesla T4 GPU and 16GB connected memory. For data preprocessing PIL and NumPy libraries were used. For model training, evaluation and results visualization were used such open-source libraries as Keras, scikitlearn, seaborne and mathplotlib.

4.3 Training parameters

All hidden layers use ReLU activation function, while the last Dense layer uses Softmax function. Kernel size for Conv2D layers is 3x3, for MaxPooling2D layers is 2x2. Kernel amount for Conv2D starts from 32 and gets doubled each layer in depth except from separately added layer, that keeps the amount the same on its turn. For choosing hyperparameters manual approach was used, and main emphasis was done on existing conclusions[5, 11, 12, 18]. While compiling model, rmsprop (Root Mean Square Propagation) optimizer,

[3] Data augmentation is generating new data from existing ones by applying different transformations.[17]

categorical crossentropy loss and accuracy metrics were used. Below there are formulas of Root Mean Square Propagation optimizer and categorical crossentropy loss.

$$b = b - \alpha \frac{\text{db}}{(\sqrt{V_{db}} + \epsilon} \qquad\qquad H(p, q) = -2\, p(x)\; log\; q(x)$$

4.4 Performance metrics

While common evaluation technique among classification tasks is top-5 error rate, in this research classification is done only over 7 classes, therefore top-1 error rate (known also as test accuracy) will be used. Test Accuracy represents correlation of amount of correctly guessed test datapoints to all test batch. Test dataset contained 10 images, while validation dataset – 20 images before augmentation. It will be observed over 200 epochs to track model's learning progress and assess its performance stability. This duration allows for sufficient training iterations without risking overfitting or excessive computational resources. Early stopping isn't used, because this research is focused on tracking and comparing model performance, especially in context of overfitting predisposition. Monitoring the curve helps to identify when the model converges and whether further training or adjustments are needed. Below there is a formula for calculating test accuracy.

$$\text{Test Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Whole test batch amount}}$$

Also, to better understand model mispredictions towards classes and specific feature types, confusion matrix will be used. It calculates True Positive Rate for each class and False Negative Rate for all other classes with respect to the current class[19]. It may be represented by formula:

$$CM_{ij} = \begin{cases} TP_i & if\ i = j \\ \sum_{k=1}^{N} \mathbb{1}_{\{y_k = i\ \text{ad}\ \widehat{y_k} = j\}} & \text{if i} \neq \text{j} \end{cases}$$

5. Obtained Results

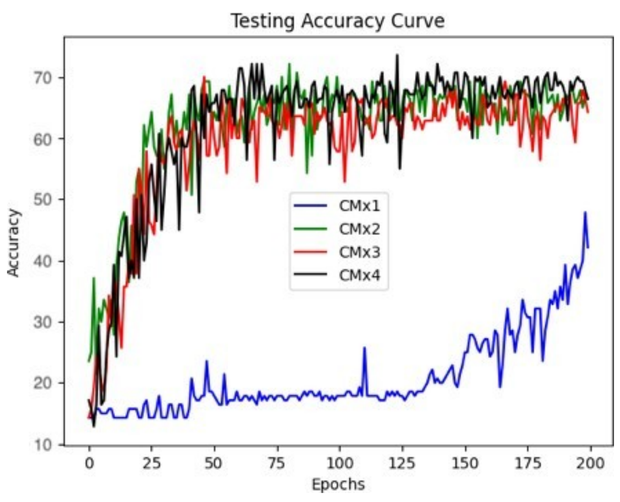


Figure 8: Test accuracy of different amount of pairs of Convolutional and Pooling Layers

Firstly, there was tested performance of different amount of pairs of Convolutional-MaxPooling layers. The highest benchmark of 72.006% was achieved by model with 4 pairs of Conv2D-MaxPool2D layers, while CMx2 and CMx3 were able to achieve almost same results.

Its notable, that even CNN with one pair of convolutional-pooling layers was able to capture some simple and middle level features
– Horn gesture (1) and Inverse Pentagram (2) significantly differ from others. At the same time it is hard for first model to distinguish between classes with similar low-level features, like Inverse Pentagram (2) and Pentagram (5), and high-level features aren't captured at all (Panks-Metalists, 3-4).

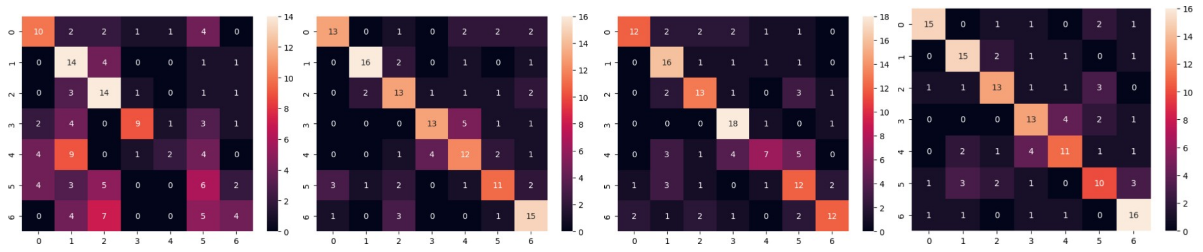
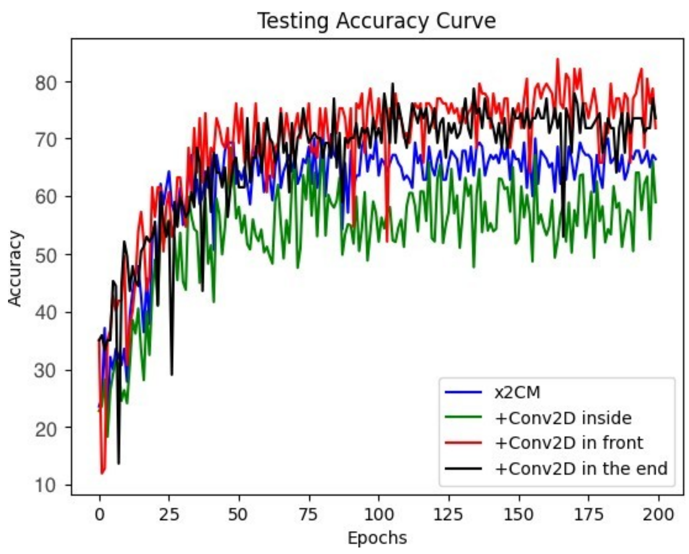


Figure 9: Confusion matrices for CMx1-4

Second Model with two pairs of feature extraction layers provides much better results: it can distinguish high-level features and similar low-level features. However, it still has a small problem in mispredicting Pentagram (5) as Inverse Pentagram (2) or Anarchy (1). Third model tends to mispredict high-level features in Metalists (3) and Punks (4), while fourth model behaves similarly to second one, despite Pentagram (5) is being misclassified more with respect to Reverse pentagram (2). Consequently, having 2 pairs of features extracting layers is enough to capture all types of features, their increasing doesn't bring any improvement, moreover it may decrease the performance. This may be caused by having too much MaxPooling2D layers, which reduce some useful but small features, that therefore can't be recognized by new kernel sliding. To address this problem, there will be only an extra Conv2D layer in different parts of 2xCM model.



It may be seen already from the testing curve that model 2 with additional Conv2D layer inside captures too much noise, while adding Conv2D layer in the end helped to reach benchmark of 83.382%. The next option to try is to increase amount of Dense layers to make model better understanding the features it extracted.

Figure 10: Testing accuracy curve change after adding Conv2D layers, in %

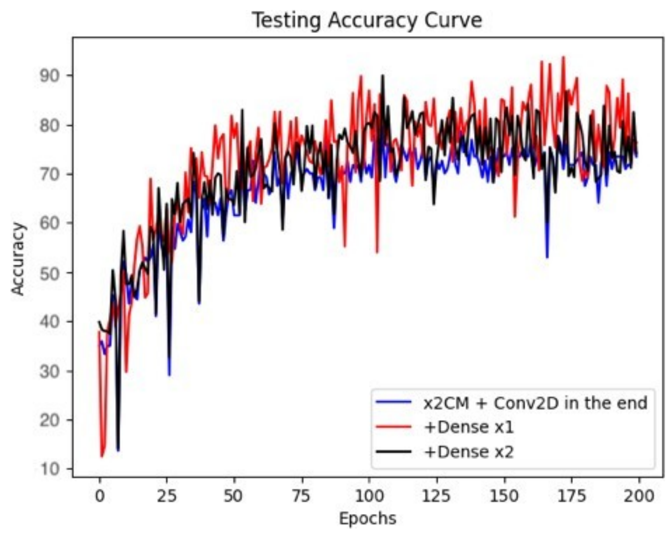


Figure 11: Testing accuracy curve change after adding Dense layers, in %

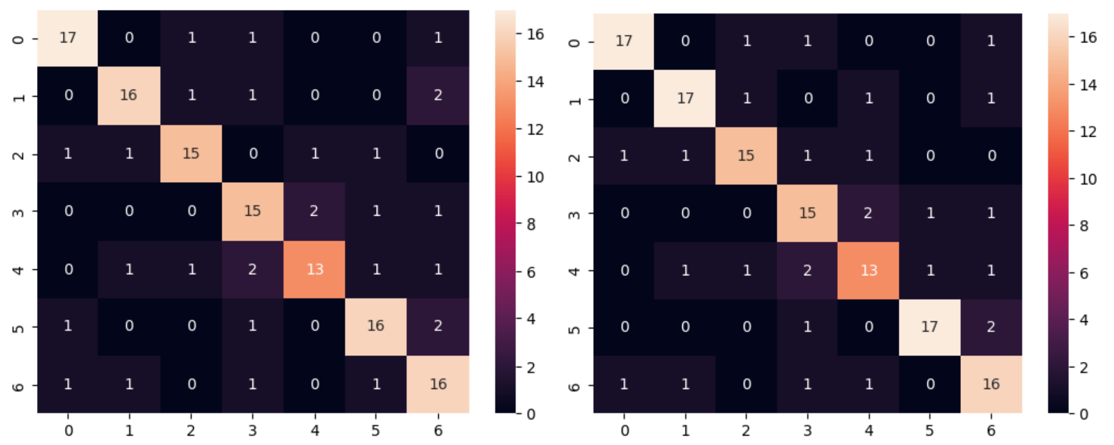


Figure 12: Confusion matrices for comparison capabilities change after adding Dense layer

While there are almost no differences in confusion matrices between adding one or two layers, however, they significantly differ from the original model by building much stronger relationship between groupds of features and classes. In particular, they were much more successful in distinguishing high-level features and objects with similar low-level features. The models with one and two additional Dense layers approached benchmarks of 89.9496% and 91.3690% respectfully.

Table 2
Comparing achieved accuracy among the topologies (CM = Conv2D + MaxPooling2D)

Model Architecture	Highest Achieved Testing Accuracy
Conv2D – MaxPooling2D – Flatten – x2Dense	49.4924%
CMx2 – Flatten – x2Dense	70.7960%
CMx3 – Flatten – x2Dense CMx4 – Flatten –	70.0933%
x2Dense	72.0060%
CMx2 – Conv2D – Flatten – x2Dense CMx2 –	83.3820%
Conv2D – Flatten – x3Dense CMx2 – Conv2D	89.9496% 91.3690%
– Flatten – x4Dense	

6. Conclusion

In this work there was studied Convolutional approach from scratch and there was practiced how modifying its parts influences model's performance. Biggest attention was brought to CNN application in classification tasks that require elusive dataset with questionable quality. In particular, the result of this research may be used in youth symbols classification tasks. There was used a self-collected dataset, that helped to provide the necessary evidence. The dataset consisted of 700 images, belonging to 7 classes, while different classes contained only low-level, low- and middle-level or all levels of possible features. Additionally, different approaches in augmentation were applied, which helped to increase model accuracy by 20% (in relative score). The obtained results from experiments showed that to extract more features, and to better distinguish similar features additional convolutional layer should be added. Adding arbitrary amount of Pooling and Dense layers helps to prevent overfitting in long perspective. However, increasing only one part of model (i.e. responsible for feature extractions or their manipulations) leads back to model overfitting. During the experiments different combinations of layers were tried, finding the most suitable ones. The results showed that model, consisting of two pair of Conv2D-MaxPooling2D layers, followed by Conv2D and three Dense layer was able to achieve more than 91% accuracy benchmark. To further optimize the results, more expanded research should be conducted, especially there should be paid high attention to Reccurent Neural Networks and existing solutions in Image Segmentation tasks[21-23].

7. References

[1] C. Ranjeeth Kumar, Saranya N, M. Priyadharshini, Derrick Gilchrist E, Kaleel Rahman M, Face recognition using CNN and siamese network, 2023, https://doi.org/10.1016/j.measen.2023.100800.

[2] V. Kubytskyi, T. Panchenko, “An Effective Approach to Image Embeddings for E-Commerce”, 2023, https://ceur-ws.org/Vol-3347/Short_5.pdf

[3] A. Azulay, Y. Weiss, “Why do deep convolutional networks generalize so poorly to small image transformations?”, 2019, https://jmlr.org/papers/volume20/19-519/19-519.pdf

[4] K. Fukushima, S. Miyake “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition.”, in “Competition and cooperation in neural nets”, 1982, https://link.springer.com/book/10.1007/978-3-642-46466-9

[5] Matthew D Zeiler, Rob Fergus, “Visualizing and understanding convolutional networks”, in “European conference on computer vision”, 2014, https://link.springer.com/chapter/10.1007/978-3-319-10590-1_53

[6] Tejaswi Potluri, Somavarapu Jahnavi & Ravikanth Motupalli, “Mobilenet V2-FCD: Fake Currency Note Detection”, 2021, https://link.springer.com/chapter/10.1007/978-981-16-3660-8_26

[7] Chiheb Chebbi, “Mastering Machine Learning for Penetration Testing”, 2018, by Packt Publishing, https://github.com/PacktPublishing/Mastering-Machine-Learning-for-Penetration-Testing

[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, “Backpropagation applied to handwritten zip code recognition. Neural Computation”, 1989, https://doi.org/10.1162%2Fneco.1989.1.4.541

[9] D. Cires¸an, U. Meier, J. Schmidhuber, “Multi-column deep neural networks for image classification”, 2012, https://doi.org/10.48550/arXiv.1202.2745

[10] Xiu-Shen Wei, Yi-Zhe Song, Oisin Mac Aodha, Jianxin Wu, Yuxin Peng, Jinhui Tang, Jian Yang, Serge Belongie, “Fine-Grained Image Analysis with Deep Learning: A Survey”, 2021, IEEE, https://doi.org/10.48550/arXiv.2111.06119

[11] A. Krizhevsky, I. Sutskever, Geoffrey E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, 2012, https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[12] K. Simonyan, A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, 2014, https://doi.org/10.48550/arXiv.1409.1556

[13] Haiguang Wen, Junxing Shi, Yizhen Zhang, Kun-Han Lu, Jiayue Cao, Zhongming Liu, “Neural Encoding and Decoding with Deep Learning for Dynamic Natural Vision”, 2016, https://doi.org/10.1093/cercor/bhx268

[14] Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li, “Empirical Evaluation of Rectified Activations in Convolution Network”, 2015, https://doi.org/10.48550/arXiv.1505.00853

[15] C. E. Nwankpa, W. Ijomah, A.Gachagan, S. Marshall, “Activation Functions: Comparison of Trends in Practice and Research for Deep Learning”, 2018, https://doi.org/10.48550/arXiv.1811.03378

[16] S. R. Dubey, S. K. Singh, B. B.Chaudhuri, “Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark”, 2022, https://doi.org/10.48550/arXiv.2109.14545

[17] Alex Hernandez-Garcia, “Data augmentation and image understanding”, 2020, https://doi.org/10.48550/arXiv.2012.14185

[18] S. Pandian, “A Comprehensive Guide on Hyperparameter Tuning and its Techniques”, 2022, https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its- techniques/

[19] Margherita Grandini, Enrico Bagli, Giorgio Visani, “Metrics for multi-class classification: an overview”, 2020, https://doi.org/10.48550/arXiv.2008.05756

[20] Wen Zhu, Nancy Zeng, Ning Wang, “Sensitivity, Specificity, Accuracy, Associated Confidence Interval and ROC Analysis with Practical SAS® Implementations”, 2010, https://lexjansen.com/nesug/nesug10/hl/hl07.pdf

[21] Song Yuheng, Yan Hao, “Image Segmentation Algorithms Overview”, 2017, https://doi.org/10.48550/arXiv.1707.02051

[22] Hebei Li, Yueyi Zhang, Zhiwei Xiong, Zheng-jun Zha, Xiaoyan Sun,“Deep Spiking-UNet for Image Processing”, 2023, https://doi.org/10.48550/arXiv.2307.10974

[23] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, Wei Xu, “CNN-RNN: A Unified Framework for Multi-label Image Classification”, 2016, https://doi.org/10.48550/arXiv.1604.04573