Family Coat of Arms and Armorial Achievement Classification

Martin Sustek, Frantisek Vidensky, Frantisek Zboril jr. and Frantisek V. Zboril
The first two authors contributed equally to this work.

FIT, Brno University of Technology, IT4Innovations Centre of Excellence,
Bozetechova 1/2, 612 66 Brno, Czech Republic,
{isustek, ividensky}@fit.vutbr.cz

Abstract. This paper presents an approach to classification of family coats of arms and armorial achievement. It is difficult to obtain images with coats of arms because not many of them are publicly available. To the best of our knowledge, there is no dataset. Therefore, we artificially extend our dataset using Neural Style Transfer technique and simple image transformations. We describe our dataset and the division into training and test sets that respects the lack of data examples. We discuss results obtained with both small convolutional neural network (convnet) trained from scratch and modified architectures of various convnets pretrained on Imagenet dataset. This paper further focuses on the VGG architecture which produces the best accuracy. We show accuracy progress during training, per-class accuracy and a normalized confusion matrix for VGG16 architecture. We reach top-1 accuracy of nearly 60% and top-5 accuracy of 80%. To the best of our knowledge, this is the first family coats of arms classification work, so we cannot compare our results with others.

Keywords: coats of arms, image classification, convolutional neural network, artificial intelligence, machine learning

1 Introduction

Family coats of arms were used as a representation of noble families. That still holds for some families nowadays, especially in Europe. Since ancient times, there was a need for distinction of groups of people, individuals or authorities all over the world. That was achieved by means of different symbols. These symbols may have special meaning in some cases; in others, they were chosen according to the environment they originate from. Afterwards, symbol creation respected fixed rules. In Europe, those rules began to arise in the eleventh century. First, symbols were displayed only on shields, later, they started to appear on other parts like helmets, crest or mantling. Although these rules have undergone many changes, they are still used in some countries today. Coats of arms with all components are called an armorial achievement. We do not distinguish between the term *coat of arms* and *armorial achievement* in this text.

Fig. 1: Two coats of arms belonging to the same class. The first image was taken from [1] and the second one was obtained from a website[1].

Despite the fact that computer vision and image processing have been applied in many areas recently, we have not observed any classification of coats of arms.

We focus on a creation of a classifier that will be able to distinguish different classes of coat of arms. Every class represents the same type of coat of arms. Coats of arms of the same family with different components are illustrated in Figure 1.

There are thousands of family coat of arms classes. More than thousand classes [2, 3] could be derived only from the Czech Republic. In order to create a classifier that could distinguish all of them is a tremendously difficult task. Trying to classify only a subset of all classes still remains a complex task. State of the art classification approach is to use one of existing convolutional neural networks architecture [4]. Large dataset is a crucial in order to obtain decent accuracy. Thus, the absence of public dataset makes the task even harder. The most of books containing coat of arms are not digitized and coats of arms are often found only in castles and churches. Therefore, our goal is to classify only 50 family coat of arms types using convnet.

2 Related work

To the best of our knowledge, this is the first family coats of arms classification work. A recognition system for individual parts of family coats of arms was introduced in [5]. This system was able to detect coat of arms in an image. Afterward, they decomposed the image into individual parts and classified them.

Convnets have many different architectures and we will describe few of the most popular ones. Some architectures were introduced during competitions such as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[2]. One of the most common architectures is VGG [6]. VGG iteratively reduces the input size and increases the number of feature maps through convolution and max

[1] http://gis.fsv.cvut.cz/zamky/genealogie/harrach/php/info.php
[2] http://image-net.org/challenges/LSVRC/

pooling layers. This can be understood as a feature extraction that is followed by fully connected layers. The Inception (GoogLeNet) architecture [7, 8] consists of inception modules. These modules have several convolutional blocks that are connected together and each block has different filter size. Each module consists of parallel branches and their outputs are concatenated. Each branch can have multiple sequentially connected blocks. The Inception architecture has far fewer parameters than the VGG architecture, and for this reason, it is faster (even though the sequential blocks slow down the computation). The ResNet architecture allows to train a really deep networks. Residual connections were introduced to deal with a vanishing gradient problem [9]. The Xception [10] network is similar to the Inception architecture with residual connections. Inception modules are replaced by depthwise separable convolutions. Depthwise convolution has fewer parameters than standard convolution because it assumes that the pattern in feature channels is the same at each position.

Fig. 2: An example of artificially generated coat of arms. The first column contains coats of arms that are considered to be the content that we want to keep in the generated image. The second column represents the style, the third is the output. The first combination is not far from the painting. The second resembles an old drawing on the wall and the last one almost looks like it is engraved in a concrete wall. Content image and style image on the first row were obtained from websites[3] and other content images were taken from books [1, 11].

[3] http://www.papirovehelmy.cz/zbozi/339-erb-rodu-bekovt.html
http://www.boskovice.cz/znak-erb-a-prapor-mesta/d-22085

3 Dataset Description

In order to train convnet, we need to collect a labeled images. We constructed our dataset using books about coat of arms [11, 1, 3]. Afterward, we explored more sources including websites and literature referenced from [11] to acquire even more data.

Our dataset consists of only 592 images belonging to 50 classes. The image distribution among classes is not balanced. For some classes, we have only 4 images, for others, we have 25 images. Whole distribution is shown in Figure 3. The need for dataset extension in order to improve classification accuracy, is evident. We decide to use data augmentation to overcome this issue. Specifically, we applied Neural Style Transfer technique introduced in [12] to generate augmented dataset.

In a nutshell, this technique takes two input images and tries to apply style from the first image and content from the second one and combine them into an artificial image. It is not as simple as described, therefore, it does not always work as expected. Extracted style sometimes reflects colors, textures or various patterns in the painting. Few examples are shown in Figure 2. We extended our dataset by 1282 artificial images.

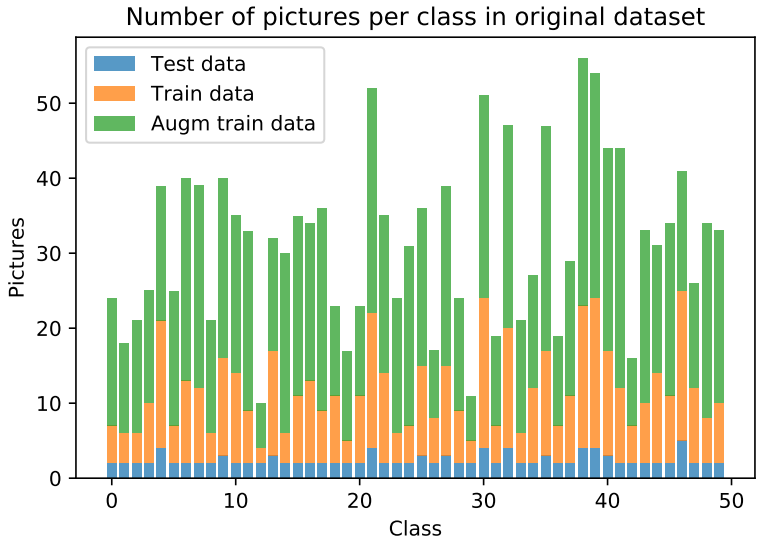


Fig. 3: The number of test, train and augmented train examples for each class in dataset. Some of augmented examples might be excluded as described above.

Table 1: Experimental result for convnet trained from scratch. The first row shows results when using basic dataset, the third shows dataset extended by images generated by Neural Style Transfer technique. The second row of the results restrict extended dataset to only those images that were generated from style and content images both occurring in training set. As a measurement, we use top-1 and top-5 accuracy on test set.

	No Dropout or Augmentation		Dropout and Augmentation	
	Top-1 ACC	Top-5 ACC	Top-1 ACC	Top-5 ACC
Basic dataset	0.2685	0.4867	**0.3844**	**0.6487**
Partially ext. dataset	0.3140	0.5123	0.3223	0.6115
Extended dataset	0.2809	0.4958	0.3471	0.5537

Table 2: Experimental results for different pretrained architectures after 30 epochs trained on extended dataset.

	30 Epochs		100 Epochs	
	TOP-1 ACC	Top-5 ACC	Top-1 ACC	Top-5 ACC
VGG16	0.5785	**0.8016**	**0.5950**	0.7933
VGG19	0.5371	0.7851	0.5867	0.7685
Xception	0.1404	0.3388	-	-
ResNet50	0.2644	0.5371	-	-
InceptionV3	0.1570	0.3553	-	-

3.1 Train and Test Data Preparation

The lack of data forced us not to use a validation set. Therefore, to be able to observe partial result, we use a test set. Using a test set as a validation set would result in incorrect outcomes and it would not be a valid approach because the dataset would not be independent. We, therefore, divide dataset into two different distinct (train and test) parts each time we run an experiment. In order to create a balanced dataset, we set a constraint that each class must have at least 2 images in the test set. Moreover, the number of test examples per class for each test set is 20% of total number of examples for classes having more than 10 examples. We created augmented dataset using two images; source and style. Source image is always taken from the dataset. We have used 10 arbitrary chosen style images that are not presented in our dataset and the rest of style images comes from our dataset. In order to prevent leaking test set into train set, we eliminate every artificial image that was generated using style or source image from our test set. Full dataset is visualized in Figure 3.

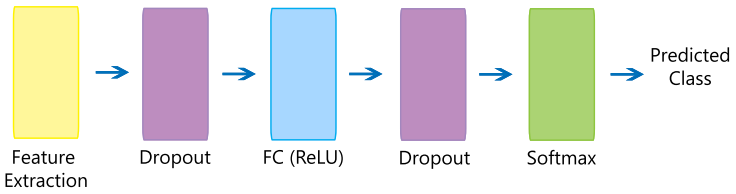


Fig. 4: Schematic illustration of our final architecture.

4 Model

We experimented with various convolutional neural networks architecture types. Training a small network from scratch did not produce sufficient results due to fast overfitting of a small dataset. We employed dropout and data augmentation [13] to fight overfitting. We augmented our dataset through image transformations. Hyperparameters defining rotation, zoom, width shift, height shift and shear were chosen arbitrarily. We also tried to increase an accuracy through extended dataset. Results after 70 epochs are shown in Table 1. Unfortunately, extended dataset seem to have little effect if any. In order to obtain better results, we decided to use pretrained model. We report results with pretrained model after 30 and 100 epochs in Table 2. VGG16 and its extended version VGG19 [6] show better results than other architectures. Therefore, we further focus on these two architectures.

4.1 Final Architectures

In our final architecture we removed top four layers from the VGG16 architectures and treated the rest of the network as a feature extractor. The feature extraction part is followed by flattening layer, dropout layer with a rate of 0.5, fully connected layer with 512 hidden neurons and ReLU activation function, dropout layer with rate of 0.25 and a softmax layer. Our architecture is visualized in Figure 4. We used one extra fully connected layers with VGG19 architecture, increased the number of hidden neurons to 1024 and set both dropout rates to 0.5.

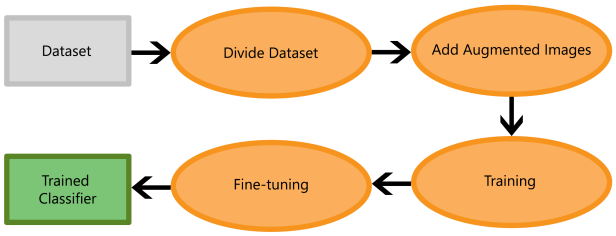
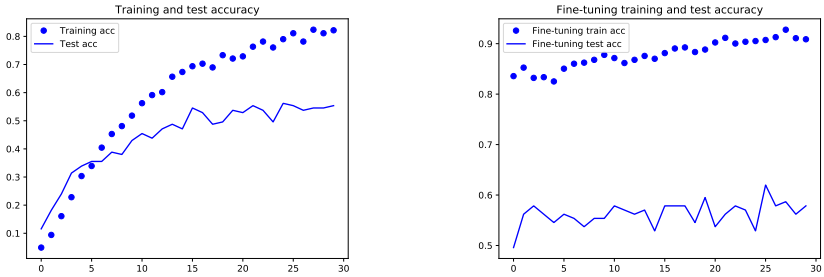
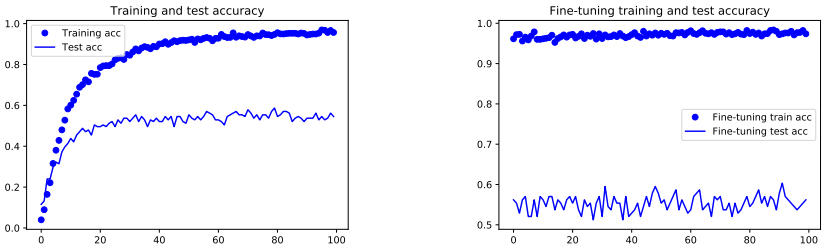


Fig. 5: Training process visualization.



(a) Training (b) Fine-tuning

Fig. 6: Accuracy during training for VGG16 trained for 30 epochs.



(a) Training (b) Fine-tuning

Fig. 7: Accuracy during training for VGG16 trained for 100 epochs.

5 Training

We use the categorical cross-entropy as a loss function for our model. Pretrained models were initialized by weights trained on ImageNet[4]. Optimization was performed using RMSProp[5] and our code is written in Keras. We set a minibatch size to 20 images and learning rate to 10^{-4} for learning and 10^{-5} for fine-tuning. Whole training process is illustrated in Figure 5. After feature extraction phase, we tried to fine-tune the network by unfreezing last 4 layers in the model that were used as a feature extraction (VGG).

6 Results and Discussion

We have already reported results for pretrained models after 30 epochs in Table 2. Figure 6 and Figure 7 show detailed accuracy progress during training and

[4] http://www.image-net.org/
[5] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

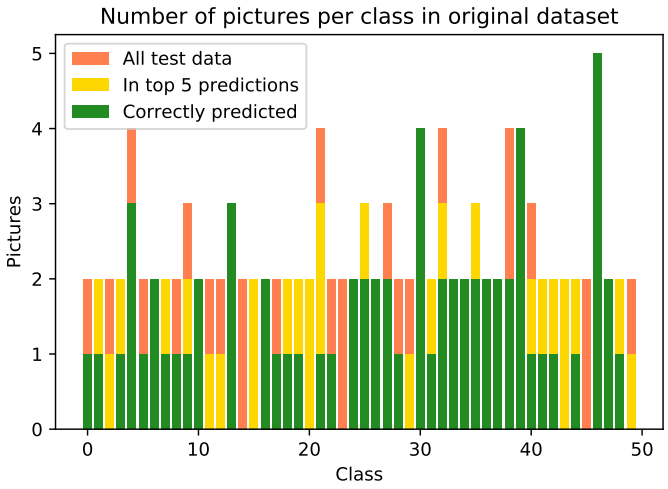


Fig. 8: Per-class accuracy for each class in test set for VGG16 after 100 epochs. Each example is colored according to the model prediction. Green color corresponds to correctly classified image. Examples for which model outputs true label in top five predictions are yellow. Remaining examples are red.

fine-tuning for 30 and 100 epochs. Training was done on extended dataset with data augmentation, however, images obtained by Neural Style Transfer did not have significant impact on accuracy. Figures suggest that there is no need for 100 epochs of training and it is enough to run fine-tuning just for few epochs. The biggest fine-tuning improvement is typically observed during the first epoch.

We also visualize per-class accuracy on test dataset in Figure 8. We get 0% accuracy for few classes but it is not crucial since there are only two examples for these classes in test set. For all classes that contain at least three images, model predicted correctly at least one example and these classes also show higher top-5 accuracy than other classes. In order to demonstrate which classes were predicted, we also provide visualization of normalized confusion matrix in Figure 9.

A typical overfitting pattern can be noticed from the test and train accuracy relation caused by our small dataset. As was mentioned before, our model deals better with classes for which we have more images. Therefore, to obtain better result, it is necessary to collect more data or to try to apply even more sophisticated image augmentation methods than we did. An alternative approach could be to use different architecture.

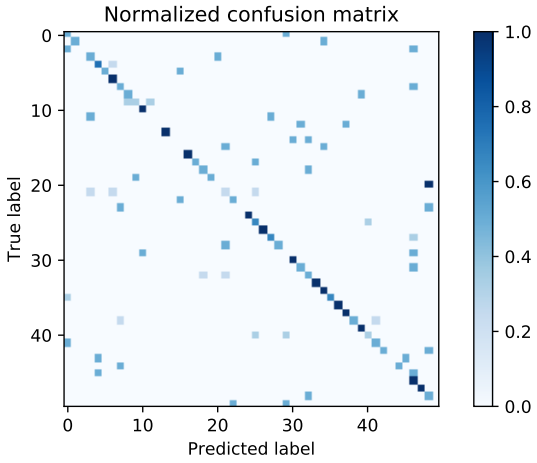


Fig. 9: Normalized confusion matrix for VGG16 after 100 epochs.

7 Conclusion

In this paper, we introduced an architecture for coat of arms classification. We experimented with Neural Style Transfer technique to augment our dataset and described conditions to enforce an independence between augmented training set and test set. Even though some of artificially generated images look realistically, our architecture does not seem to benefit from augmented dataset. We reached top-1 accuracy of nearly 60% and top-5 accuracy of 80%. Trained model could be used to distinct a small number of coat of arms classes. In order to achieve even better accuracy, it is required to collect more data. This is the first step to ultimate classifier that would be able to classify thousands of existing family coat of arms with sufficient accuracy using current machine learning techniques. Our implementation can be useful for historians or researchers.

Acknowledgment

This work was supported by the BUT project FIT-S-17-4014 and the IT4IXS: IT4Innovations Excellence in Science project (LQ1602).

References

1. Halada, J.: Lexikon české šlechty: erby, fakta, osobnosti, sídla a zajímavosti. Praha: Akropolis (1992)
2. Mysliveček, M.: Erbovník 2, aneb, Kniha o znacích i osudech rodů žijících v Čechách a na Moravě podle starých pramenů a dávných ne vždy věrných svědectví. Horizont, Praha (1997)

3. Mysliveček, M.: Erbovník, aneb, Kniha o znacích i osudech rodů žijících v Čechách a na Moravě podle starých pramenů a dávných ne vždy věrných svědectví. Horizont, Praha (1993)
4. Rawat, W., Wang, Z.: Deep convolutional neural networks for image classification: A comprehensive review. Neural computation **29**(9) (2017) 2352–2449
5. Vidensky, F., Zboril, F.: Computer aided recognition and classification of coats of arms. In: International Conference on Intelligent Systems Design and Applications, Springer (2017) 63–73
6. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
7. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 1–9
8. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 2818–2826
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
10. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. arXiv preprint (2017) 1610–02357
11. Janáček, J., Louda, J.: České erby. Oko edn. Albatros, Praha (1974)
12. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2414–2423
13. Perez, L., Wang, J.: The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621 (2017)