Visual Analysis of Heraldic Images

Petri Leskinen and Minna Tamper

Aalto University, School of Science, Finland

1 Introduction

In Finland, Heraldry is used by state, regions, municipalities, and by individual noble families. The purpose of heraldry is to give an identity to it's user that distinguishes it from others. In other words, it depicts the main characteristics of each region, municipality, or family in Finland. In this project the goal is to extract and enrich information from the coats of arms and create a catalog of them with aid of existing data that contains images of coats of arms from Finland and other Nordic countries.

The research aim of this project was studying how to extract features such as shapes, structures, colors, and textures from the images of coats of arms, and to evaluate how well this can be performed. The end result of this project is a web portal enriched with features extracted from the images for browsing and research purposes. Furthermore, during the project experiments with Image embeddings, Multi-label image classification, and Generative adversarial networks were done.

1.1 Heraldry

Heraldry is the science and the art that deals with the use, display, and regulation of hereditary symbols employed to distinguish individuals, armies, institutions, and corporations. Those symbols, which originated as identification devices on flags and shields, are called armorial bearings. Strictly defined, heraldry denotes that which pertains to the office and duty of a herald; that part of his work dealing with armorial bearings is properly termed armory. But in general usage heraldry has come to mean the same as armory.

In heraldic design[1], a *blazon* is a formal description of a coat of arms, from which the appropriate image can be reconstructed. Two examples of coats of arms with corresponding *blazons* are depicted in Fig. 1a and Fig. 1b. Figures, like the horseshoe and the boat in examples, or usually animals, flowers, human figures, buildings, etc. are called *charges*[2]. *Ordinary*[3] is a simple geometrical figure, bounded by straight lines and running from side to side or top to bottom of the shield (*escutcheon*), like for instance the two wavy horizontal bars in Fig. 1b. A coat of arms can have a *division of the field*[4], where the shields is divided into more than one area.

In the coats of arms colors or tinctures can be divided into four categories: colors, metals, stains, and furs. The colors are *azure* (blue), *gules* (red), *purpure* (purple), *sable*

[1] http://www.heraldry-scotland.co.uk/design.html
[2] https://en.wikipedia.org/wiki/Charge_(heraldry)
[3] https://en.wikipedia.org/wiki/Ordinary_(heraldry)
[4] https://en.wikipedia.org/wiki/Division_of_the_field

(black), and *vert* (green). Metals consist of two tinctures *or* (gold) and *argent* (silver). Stains are *murrey* (mulberry), *sanguine* (blood red), and *tenné* (Tawny) but these are rarely used in Nordic heraldry and considered a color variant from the main colors. Furs consist mainly of *ermine* (fur of the stoat), *vair* (coat of the red squirrel), and their variants. In addition to these colors also other colors exist. *Proper* (the colour of nature) is also used widely in coats of arms to depict, for example, the natural color of skin. *Proper*, however, is not considered a heraldic color (or tincture) and should be avoided.

(a) A horseshoe in *Azule* field, a crown above, all *Or*

(b) *Azure* a boat *Or* between two bars wavy *Argent*.

Fig. 1: Two coats of arms with corresponding *blazons*

1.2 Data

The provided data is based on the collection *Heraldica*[5] by the National Archives of Finland. The data can be browsed at *Europeana Heraldica*[6]. In total the data set provides information about 2209 coats of arms; their images and text descriptions. The image dataset included the coats of arms from Finland (625), Sweden (548), Denmark (364), and Norway (471). In the Nordic countries the coats of arms are used in provinces, regions, cities, and municipalities. These are posted at the borders and on governmental or municipal office buildings, as well as used in official documents and on the uniforms of municipal officers. Arms may also be used on souvenirs or other effects, given that an application has been granted by the municipal council.

The dataset contained 2209 images in png or gif format with approximate size of 100x115 pixels. There were 1982 coats of arms with a text description : 618 in Finnish, 821 in Swedish, 441 in Norwegian, 250 in Danish, and 25 in English. The descriptions define the *blazon*, it's shapes, colors, and possible textures with at least one language. However, for example the Finnish *blazons* often have a description in Finnish and Swedish. The data set had been divided to a training and test set, both with a size of 1004 samples.

[5] http://wiki.narc.fi/portti/index.php/Heraldica-kokoelma
[6] http://extranet.narc.fi/heraldica/

Fig. 2: Examples of *Ordinaries* and *Divisions of the field*: *Shield, Chief, Champagne, Bend, Bend Sinister, Pale, Fess, Party Per Pale, Party Per Fess*

For further research Wikimedia Commons has a collection of approx. 24 000 coats of arms[7] with descriptions in 26 languages. For some of the experiments described in this article the Nordic dataset was augmented with downloads from Wikimedia Commons.

1.3 Outline

This paper introduces the methods for shape classification, color detection, and texture identification in the sections 2 and 3 in the given order. In section 4 the results of classifications and experiments are discussed and a catalog of images extracted from the data is introduced. Lastly, section 5 describes the results and suggests possible future work.

2 Classifying shapes

The task was to classify the dataset into nine classes based on the patterns appearing in them. These patterns, *ordinaries* and *divisions of the field* with their specific names are depicted in Fig. 2. Image classification and semantic segmentation have achieve remarkable improvement in recent years using deep convolutional networks such as AlexNet[26], ZF Net[46], VGG Net[38], DenseNet[30], ResNet[18], Inception [40], and Xception[14]. The following chapters discuss the two approaches that were taken: training a simple convolutional network from scratch and transfer learning with a pre-trained network.

2.1 Classification with a simple Convolutional Neural Network

The code implementations written in Python 3.5 used Keras Deep Learning Library[4]. A neural network depicted in Table 1) with two convolutional and two fully connected layers with dropouts[39] was chosen as the classifier to be tested. The pixel size was chosen to be 32×32, and Rectified Linear Unit (ReLU)[26] activation function was used. The network architecture is similar as in the examples of classifying the MNIST dataset[11] or of data augmentation [13]. The size of the training set was relatively small, only 228 samples, where e.g. one of the classes had only 5 items. During the training the data was augmented [44] by randomly permutating the RGB-channels and adding small amounts of random scaling and rotation. Due to the nature of images, augmentation by flipping horizontally (see Fig. 2, *Bend* and *Bend Sinister*) or vertically

[7] https://commons.wikimedia.org/wiki/Category:Coats_of_arms

could not be considered. This first network resulted a precision of 58% and recall of 54%. Experiments were done by varying e.g. the input size, the number of filters and sizes of kernels of convolutional layers [3]. Similarly as with AlexNet the best results were achieved using a larger kernel size on the first convolutional layer. Expanding the CNN architecture to 3–5 convolutional layers or varying the parameters, like image size, kernel size, and learning parameters, which however did not significant improvement in the accuracy.

Layer (type)	Output Shape	Number of Parameters
Input (32, 32, 3)		
Conv2D, kernel size 9×9, ReLU	(24, 24, 16)	3904
Batch Normalization	(24, 24, 16)	64
MaxPooling2D	(6, 6, 16)	0
Conv2D, kernel size 3×3, ReLU	(4, 4, 32)	4640
Batch Normalization	(4, 4, 32)	128
Flatten	(512)	0
Dense, ReLU	(256)	131328
Dropout 50%	(256)	0
Dense, ReLU	(64)	16448
Dropout 50%	(64)	0
Dense, softmax	(9)	585
Trainable parameters:		157 001
Total parameters:		157 097

Table 1: Architecture of the CNN classifier

Fig. 3: Examples of random generated coats of arms

To expand the size of very limited data set, a Python script was written for generating images of random coats (Fig. 3). The most important random pick is the type of *ordinary*, since the data set had to have the mapping from bitmaps to the corresponding *ordinary*. The script picks a shape of shield from an existing image in the data set,

chooses two random colors for shield and ordinary, and picks a shape (Fig. 2) with varying lines of partition[8], varying contour shapes of lines were generated using SVG paths [5]. Finally the code overlays a figure picked randomly from the existing dataset, e.g. an image of an animal, building, etc. Altogether more than 20 000 training images were created.

2.2 Transfer learning using ResNet-50

Since training a complex network from scratch is a time-consuming process, one approach is *Transfer learning*: adapting the precomputed weights of large network to the new data set. For transfer learning a pretrained 50-layer Residual neural network (ResNet-50 [18]) was chosen. The same training data was used as in the case of simple CNN network.

3 Classifying colors and identifying texture

In addition to identifying shapes, another goal of the project was to identify colors and textures of different elements in the coats of arms. Both cases are discussed separately in the following chapters.

3.1 Classifying colors

In this project, the color classification of coats of arms used several different approaches in color identification. Initially the color classification was considered to be a classification task. The classification of color has been done using numerous different methods. The most common and prominent methods combine K-means [47,45] with other methods to identify for example most dominant colors [41]. In this project we decided to try out different k-means clustering methods to identify dominant colors from coats of arms. For this purpose we defined a color palette, created different clustering approaches to compare with a simple k-means algorithm, and tried out different color estimation approaches. These topics are discussed in more detail in the following sections.

Color palette In the dataset, the color palette for the coats of arms consists of 9 tinctures, as shown in the 2. The coats of arms in the dataset use mostly color and metal tinctures. The furs are rarely used and there are no references to the usage of stains. Proper is used sometimes to depict color of skin, boats, birds, goblets, and some other objects. Based on this information a color palette was created for color detection. This color palette covers colors, metals and proper. The furs and stains are ignored due to having low or nonexistent presence in the dataset. The color purpure is also removed from the color palette because of being present only in one coat of arms and because otherwise it got mixed with red often.

[8] https://en.wikipedia.org/wiki/Line_(heraldry)

Tincture	Color	Count
Azure	Blue	1433
Gules	Red	1361
Vert	Green	446
Sable	Black	576
Purpure	Purple	1
Or	Gold	2109
Argent	Silver	2358
Ermine	Fur of the stoat	3
Vair	Coat of the red squirrel	0
Proper	Color of the nature	51
Total	**9**	**9699**

Table 2: Usage of tinctures in coats of arms.

The color palette used for color detection consists in this test of 9 colors and their hex values. This color palette was created for the experiment by using the bright mid tones of each color found in the cots of arms. Also a second palette was created to test if a vast number of colors would perform better that only giving one of each. In the bigger set of colors each color had lighter tone, darker tone and a mid tone in the set. This approach, however, did not yield initially good results and ultimately the color palette of 9 tinctures was used in the color detection. This included transparent or white as it was regular background for many images especially in gif format. Without the addition, the method considered that there is silver in many images when there was only white or transparent background.

Clustering colors In order to identify colors from the coats of arms, it was decided to use pixel-based k-means and also k-d tree [9] (similarly but for image clustering [34]) to cluster the colors based on the RGB value. The RGB value of each pixel was extracted from each image using Python Imaging Library [9]. The k-means method I used, calculates the euclidean distance between color pixels to identify to which color group they belong to. In the k-d tree I created a training data from hand collected colors from random set of images. After training the classifier, it was used with the testing dataset was used.

Each coat of arms used some of the colors from the color palette. For both methods the number of colors identified from each image using multiple different metrics. For this dataset we had the number of colors defined for each coat of arms in addition to the colors. This data was used as benchmark to compare other methods could identify the number of color clusters (k). The other methods to identify the number of color clusters were elbow method [25] where k is the bend in the plot, silhouette method [33] where k

[9] https://pillow.readthedocs.io/en/latest/

is the maximum value, the Bayesian information criterion method [36] where value of k is taken using knee point analysis [48], and lastly I extracted the color names from the lemmatized text descriptions of the coats of arms and calculated the number of distinct colors mentioned there.

Once the number of color clusters were calculated using different methods, the images in png or gif format often had a white or transparent background that was interpreted as white once the image had been converted to pixels. In the case of the background, the number of clusters to be found was increased by one to take into account the background. After the identification of the colors, the mostly prominent background was removed from the list of color clusters so that it would not impact the results. However, in some coats of arms, especially the Swedish, the metal silver is often white instead of grey. Therefore, by defining the background as white it causes sometimes issues in identification of colors.

Lastly, in addition to the methods described above, another set method that was used was the super pixeling algorithms. The superpixeling algorithms were tested to see if it would be easier with less noise to identify the colors. Therefore, four different superpixeling methods were used: Felzenszwalb's method [16], Quickshift image segmentation [42], SLIC [6], and Compact watershed segmentation [31]. The methods were used to split the image of a coat of arms into small segments and afterwards a color mask was calculated on top of each segment. Afterwards, the masked versions of each coat of arms can be used in addition to the regular image with the k-means and k-d tree to cluster the colors.

Color estimation In order to identify colors, a central point coordinates of each color cluster and it's RGB value was taken and it's closeness to the colors of the color palette was estimated by converting hex values to RGB values. For the estimation three methods were selected: L1 distance and Delta E (CIE1976, CIE2000) [15,45,35]. A L1 distance [10] is used to measure a distance by calculating the sum of the absolute differences of their Cartesian coordinates as shown in the eq. 1, where $\bar{p}$ and $\bar{q}$ are vectors $\bar{p} = (p_1, p_2, ..., p_n)$ and $\bar{q} = (q_1, q_2, ..., q_n)$.

$$d_1(\bar{p}, \bar{g}) = \|\bar{p} - \bar{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i| \tag{1}$$

CIE L*a*b* (CIELAB) [15,28] is a color space specified by the International Commission on Illumination. It describes all the colors visible to the human eye and was created to serve as a device-independent model to be used as a reference. The 1976 formula is the first formula that related a measured color difference to a known set of CIELAB coordinates. This formula has been eventually succeeded by the 2000 formula because the CIELAB space turned out to be not as perceptually uniform as intended, especially in the saturated regions. This means that this formula rates these colors too highly as opposed to other colors.

Two CIELAB color distance estimation methods [37,7] were selected to be compared with one another and to the performance of euclidean distance in the given limited

[10] https://en.wikipedia.org/wiki/Taxicab_geometry

color palette. These color spaces have been added to the python colormath module [11] which was used to calculate the estimate for the nearest color.

3.2 Texture identification

The textures in the coats of arms appear in different shapes and objects. For example, one of the most frequent texture are brick textures, and fish scales. The texture detection is used frequently in different domains to identify texture from different images [8,10,23]. The methods extract features from the images and then depending on the sample size and it's features have used different trained classification methods.

In this task the sample size is originally small and therefore a complex neural network would require more data (and more work) and therefore a simpler model was implemented instead of generating more samples. In addition, the samples were in png and gif format. The original text descriptions did not contain clues about the texture of the shapes and objects that were presented in the images. In addition, the usage of furs in the blazons were limited to 3. For these reasons we decided to consider images that depicted animals, buildings, and other objects with more detailed texture as images containing texture. This would include for example fish that have been drawn scales or boats where the boards are drawn. For evaluating and training purposes we manually evaluated the images and tagged them containing texture or not containing texture.

In order to identify whether or not there are textures in the coats of arms, the Linear Support Vector Machine classifier, Logistic Regression classifier, and a Random Forest classifier was used. The data used in this task was the same data that was used for the color detection. For the given images the Haralick's image feature extraction metrics [17] were calculated and then used as features to classify the textures. In this task original images were used because initial testing indicated that the additional noise from the extracted shapes impacted the results negatively. For the tests, the data was divided into training and test sets and the models were tested with only Finnish and Swedish datasets because the Norwegian coats of arms had poorer metadata, had plenty of missing coats of arms, and little images with actual texture. All in all, the train dataset consisted of roughly 500 images and the test set of almost 400 images. In addition, the models were trained for four datasets using all data, only Finnish data, only Swedish data (with and without black and white images) to determine the impact of the number of different objects and noise on the final results.

4 Results

In this section the results are represented for classification and identification tasks. Starting with shape classification and background removal, followed by color classification and texture detection.

4.1 Classifying shapes

The following chapters represent the results of classifying the images with two chosen network setups; a simple CNN and a pretrained ResNet-50. For training the data was

[11] https://python-colormath.readthedocs.io/en/latest/

split into a training set of 80% and a test set of 20%. After the training, the results were evaluated with the 294 unseen images in the source data, e.g. the genuine images of coats of arms.

Convolutional Neural Network The simple CNN was trained using both the training set of the source data and the generated images. During the training the classifier reached an accuracy of 95% on the validation data, however when evaluated with unseen data, the accuracy remained in a range of 66–71%. The results of an evaluation are shown at Table 3 in which the columns represent the predicted cases and the rows indicate the ground truth. The overall precision was 71% and recall 68% (Table 4). By looking at the values on the left column and at the topmost row of the matrix, it can be concluded that the network does not perform quite well in cases of an empty shield. The highest precision and recall were achieved with the cases of *Bend* and *Bend Sinister*, e.g. the cases with diagonal shapes crossing the shield. The network also made misclassifications in cases of *Champagne* and *Fess*, when having a horizontal bar at the bottom or at the middle of shield.

By looking at the example coats of arms with correct classification (Fig. 4), it can be concluded that the trained network has achieved an ability to recognize the correct pattern although there might be some varying line patterns and variations to some extent in how the shape is located on the shield. Examples of incorrect classifications are shown in Fig. 5. The leftmost white-red shield on each block shows the class the image should have been classified in. Like for instance at the top, the classifier has not been able to recognize the pointed shape below the alphabet $\mathcal{R}$ as a *Champagne* pattern. Likewise on the right top the blue diagonal stripe below the animal is located too far from its default position at middle. The same holds for the too narrow horizontal bars at left bottom of the image. At the right bottom, the figures have been misinterpreted as being shape of ordinaries.

18	0	5	2	0	7	7	0	0
0	**16**	0	1	0	0	4	1	0
2	0	**35**	0	0	2	3	3	0
0	0	0	**10**	0	0	0	0	0
3	0	0	0	**6**	0	0	0	0
0	0	1	0	0	**5**	0	0	0
0	1	8	0	1	0	**30**	0	0
0	4	2	1	1	0	5	**13**	0
0	0	0	1	0	0	0	0	**8**

Table 3: Confusion matrix of the evaluation results with simple CNN using 294 images as a validation set

Fig. 4: Examples of correct classifications for classes *Chief*, *Bend*, and *Fess*

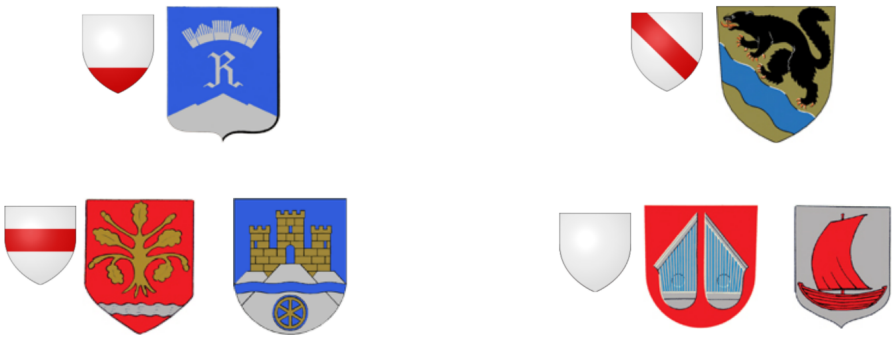

Fig. 5: Examples of incorrect classifications

Transfer learning with ResNet-50 At the first attempt the guidelines given in the reference tutorial [27] were followed. ResNet-50is trained with photographic images, and the idea of transfer learning is to shorten the training time by only training the uppermost layer, while the pretrained weights remain in the rest of the network. However, after the training the precision and recall (43%) remained low, which obviously was not a feasible outcome.

Therefore, next attempt was to train the weights of the entire network. This remarkably improved the outcome but consequently increased the training time. After the training with 32 epochs the classifier reached an accuracy of 97% on the validation data. The training loss and accuracy during the process is depicted in Fig. 6. Furthermore, when evaluated with unseen data, the accuracy was 77%, overall precision 80%, and recall 77% (Table 4). The resulting confusion matrix is shown at Table 5.

Classifier	Train accuracy	Precision	Recall	F1-score
Simple CNN	95 %	71 %	68 %	69 %
ResNet-50, training only the uppermost layer	84 %	43 %	43 %	43 %
ResNet-50, training full network	97 %	80 %	77 %	77 %

Table 4: Results of shape detection with different network architectures

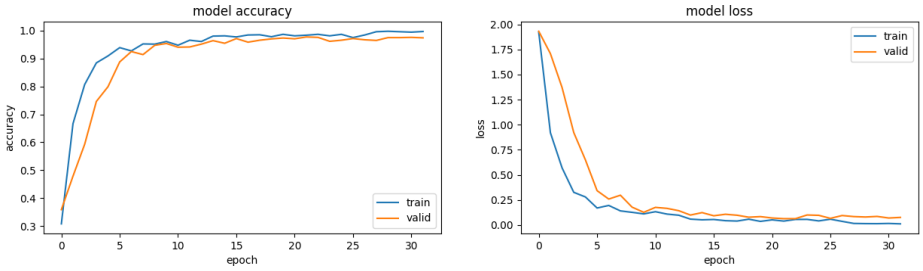

Fig. 6: Training progress of ResNet-50 with 32 epochs

Similarly as with the simpler CNN in the previous subsection, the case of an empty shield resulted the most false positives: by studying at the leftmost column of the matrix can be noticed that several coats of arms with an *ordinary* or a *division of field* were classified as empty. Fig. 7 depicts examples of these misclassifications. Presumably in images a,b, and g the pattern is too large in comparison with the patterns in the training set. In images b and c the *ordinary* gets mixed up with the motif, and the image d has different kind of visual appearance. In the image e the *fess* does not span to the edges of the shield.

In the matrix the largest non-diagonal elements with values 15 and 10 are on the third column from right. This can be interpreted that the classifier did not learn to correctly categorize the images having a horizontal division which are classes *Champagne*, *Fess*, and *Party Per Fess*. In turn, classes *Bend* and *Bend Sinister* with diagonal elements did not have any false positive classifications; all the non-diagonal elements at the fourth and fifth columns are zero. However, in a few cases images belonging to these classes were misinterpreted as empty shields (values 3 and 2 at the fourth and fifth rows of the leftmost column).

(a) (b) (c) (d) (e) (f) (g)

Fig. 7: Examples of coats of arms incorrectly classified as empty shields by ResNet-50

44	0	0	0	0	1	1	0	1
2	**18**	0	0	0	1	2	2	0
7	0	**42**	0	0	0	15	2	0
3	0	0	**13**	0	0	0	0	0
2	0	0	0	**10**	0	0	0	0
0	0	1	0	0	**7**	0	0	1
2	1	3	0	0	0	**58**	0	0
2	3	2	0	0	0	10	**25**	1
2	0	0	0	0	1	0	0	**9**

Table 5: Confusion matrix of the evaluation results with a retrained ResNet-50 using 294 images as a validation set

Fig. 8: Example of background removal

Background removal Removing the figure from the background (See Fig.8) was done with a naive Imagemagick [2] script. This script finds the shield boundary, and uses a floodfill and morphological operations to remove the pixels with same colors as found on the boundary.

The evaluation (see Table 6) was performed by sampling a random set of $N = 80$ images which had altogether 323 textual descriptions. True Positive (TP) in the evaluation indicates that a figure with its specified color mentioned in the text description has been extracted to the image as well, while False Positive (FP) stands for the areas that should have been removed, False Negative (FN) the areas that should be in the image, and True Negative (TN) for the correctly removed background parts. Overall, the simple method received a precision of 91%, and recall of 93%. After the evaluation some of the remaining errors in the dataset were fixed manually.

4.2 Color detection and texture identification

Color classification The results of the color classification were mainly positive but varied depending on the method as can be seen from the Table 7. The results have been divided based on method so that first group has comparison of the baseline and the KDTree. The baseline uses the number of colors from the dataset and clusters the

TP	FP	FN	TN
177	18	14	114

Precision	Recall	F1-score
90.77%	92.67%	91.71%

Table 6: Evaluation results of catalog images

colors using k-means algorithm. Underneath is the KDTree which performs slightly better. Underneath them are tests on how well k-means managed once the number of colors was determined using BIC, Elbow method, silhouette coefficient, and the text descriptions. From the mathematical methods the silhouette coefficient was the best and BIC the worst in this slot.

In the second slot contains the results of the Compact watershed method which did poorly. Same trend applies to SLIC, Quickshift, and Felzenswalb's methods. The results improved when using the Delta E CIE 2000 color detection method for all the groups. The precision was higher than recall for all the methods. The elbow method had the highest precision and recall of the mathematical methods in each slot whereas the prior knowledge of the number of colors yielded the highest recall. The use of Delta E CIE 2000 improved mainly the precision but had little impact on recall. When comparing the Delta E CIE 1976 and L1 distance the results are nearly the same if not worse for Delta E CIE 1976. The use of CIE 1976 drops the recall but raises the precision making little to no difference to the F1-score. In addition to F1-score, in comparison we calculated the Mathew correlation. The Mathew correlation for 500 images using Delta E CIE 2000 was for the baseline 0.75 (Normal) and for the KDTree we get 0.73.

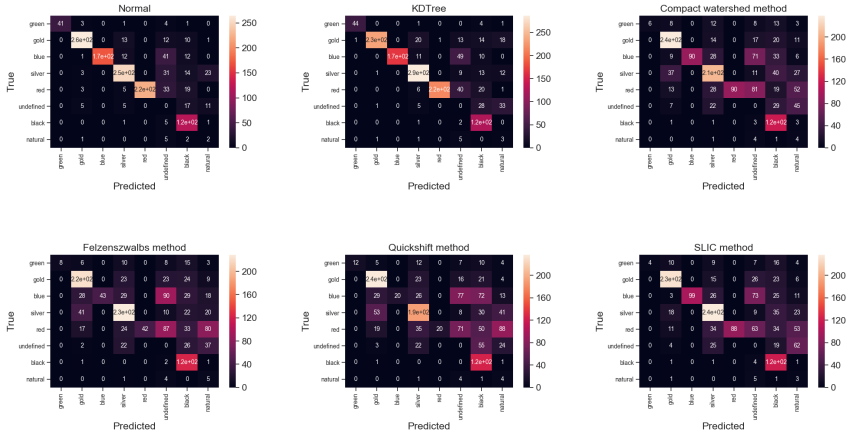

Fig. 9: Confusion matrices

Method	Precision	Recall	F1-Score
Normal	86.71 %	81.00 %	83.76
KDTree	85.19 %	81.45 %	83.28
BIC	90.79 %	68.66 %	78.19
Lemmatized	86.54 %	78.88 %	82.53
Elbow method	89.68 %	71.08 %	79.31
Silhouette coefficient	85.17 %	77.37 %	81.08
Compact watershed method	61.62 %	58.21 %	59.87
Compact watershed method with bic	63.34 %	46.56 %	53.66
Compact watershed method with elbow method	62.79 %	49.81 %	55.55
Compact watershed method with lemmatization	62.97 %	56.78 %	59.71
Compact watershed method with silhouette coefficient	64.12 %	48.30 %	55.09
Felzenszwalbs method	56.23 %	50.87 %	53.42
Felzenszwalbs method with bic	56.93 %	41.33 %	47.89
Felzenszwalbs method with elbow method	55.84 %	42.69 %	48.39
Felzenszwalbs method with lemmatization	56.27 %	49.28 %	52.54
Felzenszwalbs method with silhouette coefficient	55.21 %	41.71 %	47.52
Quickshift method	48.75 %	45.72 %	47.19
Quickshift method with elbow method	48.41 %	39.29 %	43.38
Quickshift method with lemmatization	49.51 %	45.80 %	47.58
Quickshift method with silhouette coefficient	48.94 %	40.27 %	44.19
Quickshift method with bic	48.24 %	38.38 %	42.75
SLIC method	63.63 %	60.26 %	61.90
SLIC method with bic	65.04 %	48.30 %	55.43
SLIC method with elbow method	63.12 %	50.79 %	56.29
SLIC method with lemmatization	63.14 %	57.83 %	60.37
SLIC method with silhouette coefficient	63.53 %	50.64 %	56.36

Table 7: Results of color detection using colormath library's Delta E CIE 2000 for 500 images using vast number of methods and adding gifs without color purple.

In order to analyze the errors, we generated a confusion matrix to analyze the errors as shown in Fig. 9. The figure has 6 matrices, one for each method and the heraldic colors on both axes. There is also undefined for missing colors. If a color was not identified, the color has been tagged as undefined. From the plots we can see the distribution of identified colors and what were often hard to identify (the unidentified slot). The confusion matrices show that mainly the colors are identified correctly, but in few instances they are mixed. Examples of different clusters are shown in Fig. 10, Fig. 12, and Fig. 11. As shown in the confusion matrices and in color clusters, silver mixes often with blue and green whereas the red mixes often with natural color, silver, and gold. The color clusters sometimes contain other colors also as one image contained many other colors in the borders of the shapes in addition from the textures of the shapes. Therefore each image contains more colors than what meets the eye but other colors are often scattered to the borders and textures of shapes in small numbers.

Fig. 10: Different red clusters

Fig. 11: Different silver clusters

Fig. 12: Different blue clusters

Texture identification The texture detection results were in general not satisfactory. The results can be seen in the Table 8 and have been calculated for original images. The results have been divided based on dataset and were calculated with multiple different classifiers. In the table the best results for each dataset and the classifier responsible are shown with precision, recall, F1-score, accuracy, and the number of test documents (N). The results were better for the bigger dataset that was trained and tested with the Finnish (FI) and Swedish (SV) coats of arms. The Finnish coats of arms were good quality color images. The Swedish coats of arms also included a number of black and white images. These images had a lot of noise and were black and white photocopies from the source material. To determine if the noise has an impact on the results of the Finnish and Swedish dataset (FI,SV), the dataset was split into Finnish (FI), Swedish (SV: Noisy), and to a dataset from which the black and white images were removed (SV: Clean). Afterwards, models were trained for the three new dataset and the test dataset was similarly split to test each model. The Swedish datasets (Clean, Noisy) were tested with only Swedish coats of arms[12] and the Finnish similarly with only Finnish coats of arms.

Method	Train/test dataset	Precision	Recall	F1-Score	Accuracy	N
Linear SVM	FI,SV	59 %	52 %	53 %	52 %	385
Random Forest Classifier	FI	63 %	54 %	52 %	54 %	176
Linear SVM	SV: Noisy	68 %	44 %	44 %	44 %	280
Random Forest Classifier	SV: Clean	61 %	53 %	54 %	51 %	280

Table 8: Results of texture detection for original images with the best classifier.

As can be seen from the Table 8 the noise in the Swedish coats of arms training dataset can impact the overall results of the identification task. The F1-score is higher for the cleaned dataset whereas the noisy had a poorer performance. The precision was the highest for the noisy Swedish and the Finnish datasets whereas the recall was highest for the cleaned Swedish dataset and the mixed dataset of Finnish and Swedish coats of arms. The removal of noisy images from the Swedish dataset resulted in the drop of precision compared to the noisy dataset but it improved the recall in addition to the f1-score.

4.3 Catalog for Figures in Heraldic Images

Constructing a web portal[13] for cataloging the coats of arms was also a part of the project. The portal catalogs the coat of arms by country and its municipality, city, region, or other metadata that was provided in the original dataset. This metadata is coupled with the information extracted from the images (shapes, colors, texture) in the portal.

[12] The black and white coats of arms were not removed from this testing dataset.

[13] https://cestlavieclaire.github.io/heraldry/

The extracted data was collected using the methods described in the earlier sections and for the most of the dataset. In addition, the portal has visualization tool that shows the coats of arms on a map. These applications on the metadata are explained in the following subsections in more detail.

Catalog The catalog pages combine the figures of the coats of arms, and corresponding textual descriptions. The figures were the *charges*[14], e.g. animals, human figures, building etc. extracted from the image database. Altogether the dataset consists of 1611 images in png format, 18 generated html pages to easen browsing the data, and a csv formatted list.

Examples of image catalog data are depicted in Figure 13. As an example, in the middle the coat of arm of *Keski-Suomi* has three descriptions: *shield: silver; capercaillie: black; its beak, eye, and claws: red*. The shapes can be seen without the background by clicking the name.

Fig. 13: Samples from the image catalog

Map The map page displays the coats of arms on a map as can be seen in Figure 14. In order to map the coats of arms, the datataset metadata was enriched using a service provided by Google's Geocoding API[15] to identify coordinates for each city, municipality, or region. The service took a name and returned coordinates for that place. Based on the coordinates on the locations of the places the coats of arms were placed on a OpenStreetMap map with the help of OpenLayers in JavaScript.

[14] https://en.wikipedia.org/wiki/Charge_(heraldry)
[15] https://developers.google.com/maps/documentation/geocoding/intro

Fig. 14: Mapping coats of arms.

4.4 Embedding coats of arms

Image embedding or image featuration refers to a set of techniques for dimensional reduction and building image similarity models by constructing lower dimensional numeric vector representations from the image dataset. Furthermore, these representations are used e.g. for clustering, similarity search, and recommendation. An experiment of generating a embedding for the coats of arms was made using image featuration package *pic2vec*[16]. The two chosen image classifiers were fast-running *SqueezeNet* [20] and state-of-art network *Xception* [14]. For this experiment, the dataset was augmented with approx. 19 000 coats of arms downloaded from the Wikimedia Commons dataset. The similarity of Finnish coats is visualized in Figure 15. For this visualization the dimensionality was reduced in two steps; first using Truncated Singular Value Decomposition (TSVD)[17] to reduce the 512 dimensional embedding vectors into 32 dimensions, which was further projected into two dimensions using T-distributed Stochastic Neighbor Embedding (t-SNE)[18] [29]. In the figure the clustering seems to be mostly dominated by the main colors which might be explained by reduction from 512 dimensions to two-dimensional space.

Figure 16 depicts results for image similarity search. On each row, the leftmost image is the query item, the other five are the search results in the order of descending similarity value. Image recommendations were calculated using pairwise cosine similarity in the 512-dimensional embedding space. Unsurprisingly, common themes like lions, trees, or fish provide plausible results. In the middle row the strongly geometric

[16] https://github.com/datarobot/pic2vec
[17] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html
[18] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

image with diagonal lines returns results with lines having the same direction. The bottom row shows image search results with a ship, where three of the results also have a ship, while the two other were presumably chosen by color similarity.

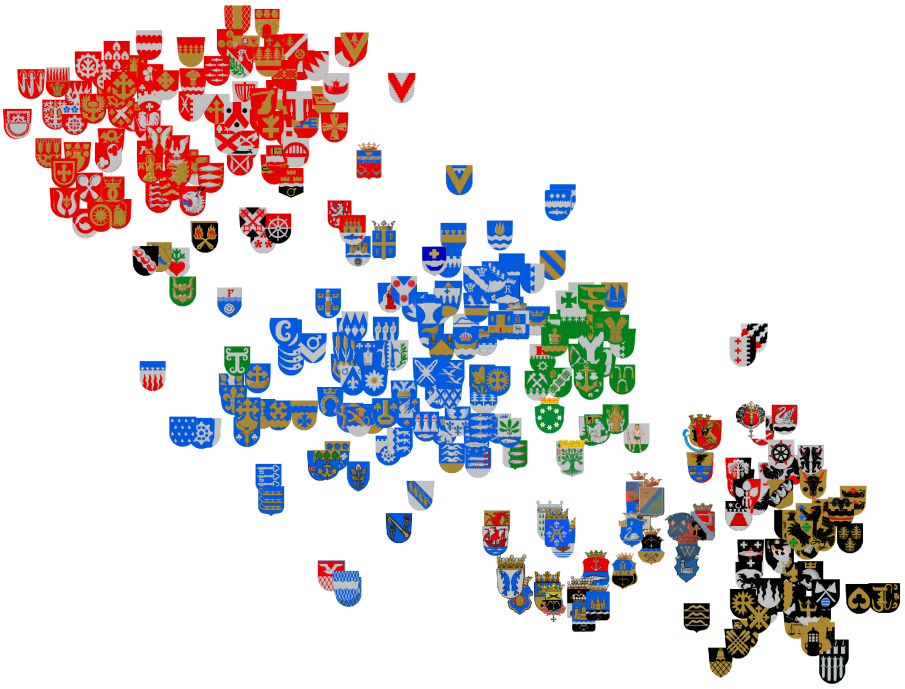

Fig. 15: Coats of arms of Finnish municipalities mapped into a two-dimensional space by similarity

4.5 Multi-label image classification

An experiment of transfer learning with a pretrained classification network was done to test recognizing the *charges*. The network setup was based on GitHub project[19] Multi-label image classification[20] with image classifier Inception v3 [40]. The training data was downloaded from the Wikimedia Commons. From the *blazon* descriptions seven common *charges* were picked up: the keywords were English, French, and Spanish words for *castle*, *crown*, *eagle*, *flower*, *lion*, *tower*, and *tree*. The query resulted with altogether 7548 images. 5747 of the images had only one class, 1492 had two, 261 three, 45 four, and 3 five classes.

[19] https://github.com/BartyzalRadek/Multi-label-Inception-net

[20] https://towardsdatascience.com/multi-label-image-classification-with-inception-net-cbb2ee538e30

Fig. 16: Examples of inferring image similarity, on each row the leftmost image is the search item, and the other five are query results.

The model was retrained in Aalto Triton service[21]. The dataset was separated into a training set of 80% of set size, 10% for testing, and 10% for validation. Training was repeated a couple of times with varied hyperparameters. At best the final accuracy was 88.3% for training, and 87.1% for validation with a cross entropy of 0.3121. This value for cross entropy indicates that the classifier is not absolutely confirmed about the decision. Examples of classified images are depicted in Figure 17. The tables next to images show the probabilities, with values from 0.0% to 100.0%, that the mentioned item appears in the image.

In this experiment only seven *charges* were chosen from *blazons* in three languages. This could be studied further by using a larger set of classification classes, and by using a domain specific vocabulary (e.g. HERO - Ontology of Heraldry[22]) to extract the terms from a larger set of languages.

[21] https://scicomp.aalto.fi/triton/

[22] http://dev.finto.fi/hero/en/

lion	34.18
tower	30.58
flower	17.70
castle	16.19
crown	15.23
eagle	10.42
tree	7.55

lion	52.97
tower	22.44
tree	21.29
crown	20.10
eagle	17.88
flower	15.94
castle	13.84

Fig. 17: Examples of Multi-label classification

4.6 Generating random coats of arms

Variational Autoencoders (VAE)[1] and Generative Adversarial Networks (GAN)[22,32], have been used to generate photorealistic images based on a training set of real images. Like for instance a variational auto-encoder could provide a way to e.g. a latent space interpolation between two coats of arms[19].

As an experiment a deep convolutional GAN (DCGAN)[24] was tested on the dataset, examples of resulting images are depicted in Fig. 18. Due to probably small dataset size or a too short training time the results seem to lack the details, and appear as artistic sketches of coats of arms. The result set also had limited variation, e.g. similar images kept repeating in the result set.

5 Discussion, Related Work, and Future Research

Due to relatively small dataset size, image captioning[12] was not researched, although coats of arms with corresponding *blazons* would have provided an interesting domain of research. In addition, the coats of arms were not divided for Norway and Denmark to municipalities, provinces, and bishoprics. The coats of arms cannot be grouped consistently for each country. This issue could be fixed by linking the coats of arms to Wikidata or Wikipedia for gather more data that can make it possible to group them better. Also, linking of coats of arms could be used to create tools and intelligent applications for studying purposes.

Shape classification Two neural networks of different architectures were used for the shape classification. The retrained ResNet-50 received precision of 80% and recall of 77%, while the simpler convolutional neural network had precision of 71% and recall of 68%. Generally, there were cases in which both the classifiers had similar problems: on one hand to distinguish between a *Ordinary* and a image motif and on the other hand to recognize a shape that deviates too much from the patterns in the training data. After we had started our work, a related article was published [43]. It had the focus on Czech heraldry, generally with more complex elements in the design, e.g. helmets, torses, and mantlings. It dealed with a locating the position of coats of arms in photographies, while our data had clearly cropped coats.

Fig. 18: Images generated by DCGAN

Color classification The color classification performed well but it could have performed better. The CIE2000 color detection method improved the results in addition to increase in images. For all images the F1-score reached 75% which is still worse in comparison [21]. The problems of the color classification are in the color detection and the fact that the dataset contains gray images. Often the color detection ends up guessing colors and just picks a shade of gray closest to each color. What could be done better is to remove the gray images and replacing them with their colored versions that can be found most of the times from Wikimedia Commons. In addition, the dataset did not contain almost any furs or stains and it could have been interesting to study how to identify these features.

The model used for color detection also had some limitations. The color identification methods impacted somewhat the results of the tests. Also, the image contains more colors than what meets the eye but other colors are often scattered to the borders and textures of shapes in small numbers. However, tests with higher k-number do not improve results because the end results contained more different shades of grey which would include also the pixels of other colors. This means that when trying to identify for example the red tongue of a goat, the blues are added to other color clusters and even increasing the k value and picking up unique colors doesn't help. The image contains other colors also and those colors are getting clustered and form other color clusters

which increases the number of wrongly identified colors. Identifying shapes from the images and creating layers that are colored based on the dominant or average color on the other hand also sometimes gets cluttered because of the textures and borders of shapes. Currently, using the image segmentation methods produce worse results than the KDTree. The KDTree results got better when more shades of the main colors were added into the color palette that is used to identify colors. The baseline method and others got worse, the more there is diversity in shades of the main colors in the color palette.

Texture detection The size of the dataset also impacted the quality of the texture detection. The coats of arms have a vast number of different animals, buildings, and other objects on them. In order to have a successful texture detection method, more coats of arms with different kinds of objects with and without texture. Only a handful of birds or fish with and without texture is not enough to train the model to determine better the existence of texture in the object. The fish may have fins and the birds feathers but these animals come in different shapes and sizes (and some with outline around the object), making it hard to train the model to determine texture from the objects. Also, at times the image quality was an issue as many of the Swedish coats of arms were black and white pictures that looked like they were scanned from a mispositioned sheet of paper. The removal of such images improved the texture detection notably.

References

1. Generating Large Images from Latent Vectors. http://blog.otoro.net/2016/04/01/generating-large-images-from-latent-vectors/, accessed: 2018-03-06
2. Imagemagick: image processing library. http://www.imagemagick.org, accessed: 2018-03-05
3. Keras documentation, convolutional layers. https://keras.io/layers/convolutional/, accessed: 2018-03-05
4. Keras: The Python Deep Learning library. https://keras.io/, accessed: 2018-03-05
5. MDN web docs: Paths. https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Paths, accessed: 2018-03-06
6. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., et al.: Slic superpixels compared to state-of-the-art superpixel methods. IEEE transactions on pattern analysis and machine intelligence 34(11), 2274–2282 (2012)
7. Autiokari, T.: Evaluation of the the CIE Color Difference Formulas. https://web.archive.org/web/20080705054344/http://www.aim-dtp.net/aim/evaluation/cie_de/index.htm (2007), accessed: 2019-01-01
8. Barata, C., Ruela, M., Francisco, M., Mendonça, T., Marques, J.S.: Two systems for the detection of melanomas in dermoscopy images using texture and color features. IEEE Systems Journal 8(3), 965–979 (2014)
9. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM 18(9), 509–517 (1975)
10. Bovis, K., Singh, S.: Detection of masses in mammograms using texture features. In: Proceedings 15th International Conference on Pattern Recognition. ICPR-2000. vol. 2, pp. 267–270. IEEE (2000)

11. Brownlee, J.: Handwritten Digit Recognition using Convolutional Neural Networks in Python with Keras. https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/, accessed: 2018-03-05
12. Brownlee, J.: How to Develop a Deep Learning Photo Caption Generator from Scratch. https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/, accessed: 2018-03-08
13. Chollet, F.: Building powerful image classification models using very little data. https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html, accessed: 2018-03-05
14. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. arXiv preprint pp. 1610–02357 (2017)
15. Connolly, C., Fleiss, T.: A study of efficiency and accuracy in the transformation from RGB to CIELAB color space. IEEE Transactions on Image Processing 6(7), 1046–1048 (1997)
16. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International journal of computer vision 59(2), 167–181 (2004)
17. Haralick, R.M., Shanmugam, K., et al.: Textural features for image classification. IEEE Transactions on systems, man, and cybernetics (6), 610–621 (1973)
18. 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. pp. 770–778 (2016)
19. Hou, X., Shen, L., Sun, K., Qiu, G.: Deep Feature Consistent Variational Autoencoder. In: Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on. pp. 1133–1141. IEEE (2017)
20. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡ 0.5 MB model size. arXiv preprint arXiv:1602.07360 (2016)
21. Kakumanu, P., Makrogiannis, S., Bourbakis, N.: A survey of skin-color modeling and detection methods. Pattern recognition 40(3), 1106–1122 (2007)
22. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: Proc. International Conference on Learning Representations (ICLR) (2018)
23. Kim, K.I., Jung, K., Kim, J.H.: Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(12), 1631–1639 (2003)
24. Kim, T.: A tensorflow implementation of "Deep Convolutional Generative Adversarial Networks". https://github.com/carpedm20/DCGAN-tensorflow, accessed: 2018-03-06
25. Kodinariya, T.M., Makwana, P.R.: Review on determining number of Cluster in K-Means Clustering. International Journal 1(6), 90–95 (2013)
26. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
27. Kumar, S.: Tutorial Keras: Transfer Learning with ResNet50. https://www.kaggle.com/suniliitb96/tutorial-keras-transfer-learning-with-resnet50, accessed: 2019-05-14
28. Luo, M.R., Cui, G., Rigg, B.: The development of the CIE 2000 colour-difference formula: CIEDE2000. Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur 26(5), 340–350 (2001)
29. Maaten, L.v.d., Hinton, G.: Visualizing data using t-SNE. Journal of machine learning research 9(Nov), 2579–2605 (2008)

30. Majumdar, S.: Fully connected densenet for image segmentation (2018), https://github.com/titu1994/Fully-Connected-DenseNets-Semantic-Segmentation, [Online; accessed 1-February-2018]
31. Neubert, P., Protzel, P.: Compact watershed and preemptive slic: On improving trade-offs of superpixel segmentation algorithms. In: Pattern Recognition (ICPR), 2014 22nd International Conference on. pp. 996–1001. IEEE (2014)
32. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
33. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics 20, 53–65 (1987)
34. Saegusa, T., Maruyama, T.: An fpga implementation of k-means clustering for color images based on kd-tree. In: 2006 International Conference on Field Programmable Logic and Applications. pp. 1–6. IEEE (2006)
35. Sanda Mahama, A.T., Dossa, A.S., Gouton, P.: Choice of distance metrics for RGB color image analysis. Electronic Imaging 2016(20), 1–4 (2016)
36. Schwarz, G., et al.: Estimating the dimension of a model. The annals of statistics 6(2), 461–464 (1978)
37. Sharma, G., Wu, W., Dalal, E.N.: The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur 30(1), 21–30 (2005)
38. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
39. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1), 1929–1958 (2014)
40. 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. pp. 2818–2826 (2016)
41. Tominaga, S.: Color classification of natural color images. Color Research & Application 17(4), 230–239 (1992)
42. Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. In: European Conference on Computer Vision. pp. 705–718. Springer (2008)
43. Vidensky, F., Zboril, F.: Computer aided recognition and classification of coats of arms. In: International Conference on Intelligent Systems Design and Applications. pp. 63–73. Springer (2017)
44. Wang, J., Perez, L.: The effectiveness of data augmentation in image classification using deep learning. Tech. rep., Technical report (2017)
45. Wu, M.N., Lin, C.C., Chang, C.C.: Brain tumor detection using color-based k-means clustering segmentation. In: iih-msp. pp. 245–250. IEEE (2007)
46. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. pp. 818–833. Springer (2014)
47. Zhang, C., Xiao, X., Li, X., Chen, Y.J., Zhen, W., Chang, J., Zheng, C., Liu, Z.: White blood cell segmentation by color-space-based k-means clustering. Sensors 14(9), 16128–16147 (2014)
48. Zhao, Q., Xu, M., Fränti, P.: Knee point detection on Bayesian information criterion. In: Tools with Artificial Intelligence, 2008. ICTAI'08. 20th IEEE International Conference on. vol. 2, pp. 431–438. IEEE (2008)