Understanding Convolutional Neural Networks

Kameswari Devi Ayyagari
Faculty of Computer Science
Dalhousie University
devi.ayyagari@dal.ca

Martin Gillis
Faculty of Computer Science
Dalhousie University
martin.gillis@dal.ca

Harshit Lakhani
Faculty of Computer Science
Dalhousie University
harshit.lakhani@dal.ca

K. C. Pramir
Faculty of Computer Science
Dalhousie University
pr260461@dal.ca

Abstract

Convolution neural networks have been successfully trained to classify images. Through this project, we aim to understand the properties of convolutional neural networks in the context of classifying shapes in 200×200 images. More specifically, this study investigates the questions "Are CNNs implicitly scale and translation invariant? If not, can we design network architectures to be scale and translation invariant?". We show that by using training regularization strategies like data augmentation, and by intentionally choosing network parameters like type of pooling and filter sizes, we can train neural network models that are scale and translation invariant. We achieve accuracies of 97% and 81% for the test sets of scale invariance and translation invariance datasets respectively. We also explore several visualization strategies in an attempt to explain our models' decisions.

1 Introduction

The robustness and use of deep learning algorithms have found many applications today from image classification, objection detection, sound and video analyses, and others. Despite their ubiquitous usage and application in different fields, the relevance and role of different network parameters and augmentation techniques in the context of the dataset being used and the problem being solved is often not well understood and glossed over when choosing a network architecture. In this study we set out to understand the impact of tuning these parameters and techniques on the behavior of convolution neural networks. We approach this by trying to train neural network architectures with two desirable properties — scale and translation invariance. We generate datasets that are designed to test these 2 properties: study the effects of different pooling techniques, filter sizes, and augmentation strategies in the context of both the datasets; and successfully train scale invariant and partially translation invariant neural network models.

2 Synthetic Data Generation

We generated two sets of synthetic data, one each for scale and translation invariance experiments. Each dataset was partitioned into 9000, 900, and 900 train, valid, and test splits with 200×200 images, equally distributed across 9 different classes (i.e., triangle, square, star, pentagon, hexagon, heptagon, octagon, nonagon, and circle) in each split. For the scale invariance study, testing images had shapes that differ from training and validation such that, the testing dataset had shapes with a different scale range. Similarly, for the translation invariance study, testing images had shapes that

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

differ from training and validation such that, the testing dataset had shapes with a different location range.[1]

3 Neural Network Architecture

3.1 Loss Function and Optimizer

We used the multi-class cross entropy loss function through Pytorch's $nn.CrossEntropyLoss$ function to compute the loss between the model's raw unnormalized dense layer output vector and ground truth string labels. We conducted all our experiments using PyTorch's $Adam$ optimizer with a learning rate of 0.0001, unless explicitly specified.

3.2 Pooling

Pooling is often used in a convolution neural network to reduce the dimensions of the feature maps and summarize local features. In order to better understand the kind of pooling that is best suited for our scenario, we convolved several images with the edge filter $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$, and applied maxpooling and average pooling with a filter size of 2×2 on the convolved image. The results are shown in Figure 1. We notice that we lose some pixels that define the shape of objects when we apply average pooling. We reason that our images have sharp features and average pooling is smoothing these features out. Maxpool on the other hand, greedily grabs the pixel with the highest intensity for every maxpool block, retaining the sharp features of the polygons.

To study the behavior of the max-pooling operation in the context of translation invariance, we apply a maxpooling filter of 2×2 over a grid of [7, 8] with "active" blocks of size 2×2 over different locations in the grid. We observe that maxpooling operation, across locally translated active blocks, retains the summary of local features, as shown in Figure 1 and note that the amount of "locality" is determined by the size of the maxpool filter.

4 Baseline Model

We started by training a neural network with 4 convolutional layers, each with filter sizes of 3×3, followed by a ReLU activation function, followed by a maxpooling layer with a filter size of 2×2 for the first three maxpool layers and a filter size of 4×4 for the last maxpool layer, and 1 dense layer on unnormalized images on the scale invariance and translation invariance datasets. No augmentation strategies were applied during the training procedure. This model overfit on the validation set within 25 epochs. We normalized the images, but failed to successfully train this model.

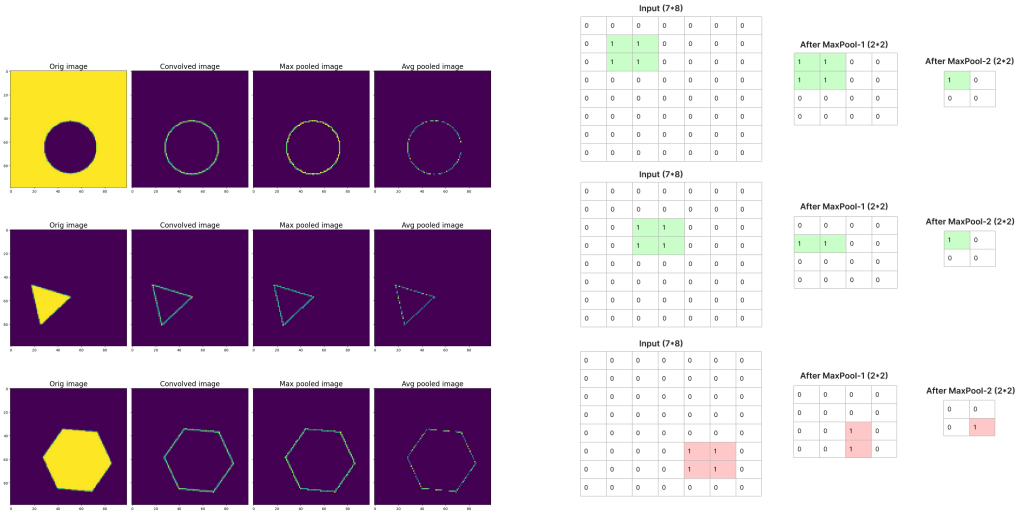
4.1 Analysis

We hypothesized that the variation in data and the amount of data the model is training on is insufficient and that the model was memorizing the data points without learning any useful features.

After several failed attempts to train the above model, we decided to use our images with a model trained using the Fashion MNIST dataset [2]. This model trained successfully using our images resulting in a training and validation accuracy of 87% and 86%, respectively. After comparing the network architectures, we identified the following two main differences that we hypothesized were strongly influencing performance:

1. Rotation Augmentation: We only have 9000 samples, each of size 200×200 distributed equally across nine classes. We hypothesize that adding augmentation will add variation to the data and simulate larger datasets, prompting the model to move away from memorization, steering it towards learning useful features for classification.

[1] Source code from [1] was adopted to generate synthetic datasets. For a visual representation of images, see Appendix Figures A.1.1 and A.2.1.



(a) (b)

Figure 1: Visualization of convolution and pooling. Left-to-right: a) Columns: 1, raw images; 2, convolved image with an edge filter; 3, max pooled image with a filter size of 2×2 applied on convolved image; 4, average pooled image with a filter size of 2×2 applied on convolved image. Note: All figures are resized to the size of the pooled images for the purpose of visulaization. b) Partial invariant nature of maxpooling layers.

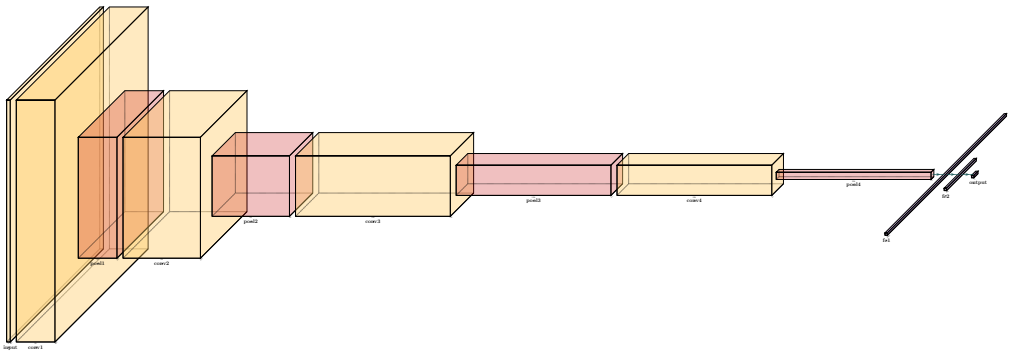


Figure 2: Neural network architecture. Left-to-right: input image $\rightarrow$ $\text{conv}_1(\text{filter}_1) \xrightarrow{ReLu}$ $\text{maxpool}_1(\text{filter}_2) \rightarrow \text{conv}_2(\text{filter}_1) \xrightarrow{ReLu} \text{maxpool}_2(\text{filter}_2) \rightarrow \text{conv}_3(\text{filter}_1) \xrightarrow{ReLu}$ $\text{maxpool}_3(\text{filter}_2) \rightarrow \text{conv}_4(\text{filter}_1) \xrightarrow{ReLu} \text{maxpool}_4(\text{filter}_3) \rightarrow \text{fc}_1 \xrightarrow{ReLu} \text{fc}_2 \rightarrow$ output.
Filters: filter_1: size = (3, 3), stride = 1, padding = 1; filter_2: size = (2, 2), stride = size, padding = 0; filter_3: size = (4, 4), stride = size, padding = 0.

2. Number of Dense layers: For the single dense layer, we are implicitly enforcing a linear relationship between the 4064 features from the convolution layers and the output classification vector. We hypothesized that introducing two layers may allow the model to learn non-linear relationship between the feature and classification vectors.

Incorporating these findings, we trained a neural network architecture that consisted of four convolution, ReLu activation, and pooling units followed by two dense layers. This model has a total of 831,945 parameters, all of which are trainable and is depicted in Figure 2.

5 Towards Scale Invariance[2]

Through this experiment, we aim to explore scale invariance in convolution networks and aim to design a scale invariant convolution neural network.

5.1 Data Augmentation

As noted earlier, data augmentation adds variability and volume to the training dataset and acts as a regularizer during training.

5.1.1 Hypotheses

We expect that training the model using rotation, flip, and affine augmentations will successfully classify shapes in the scale range that the model was trained on, but will misclassify images across varying scales in the test set. We expect zoom-in and zoom-out augmentations to expose the model to more diversely scaled shapes during training and consequently, boost the performance on test set. However, we consider zoom-in and zoom-out augmentations to circumvent our experimental design and strive to design an inherently scale invariant network architecture.

5.1.2 Experimentation and Analysis

Training the model described in Figure 2 with rotation augmentation resulted in training accuracy of 99.0%, validation accuracy of 99.8% and test accuracy of 78.7% (Table 1, Entry 1 vs 2). This model misclassified very few shapes from the scale range the model was trained on.

At this stage, we make an observation about our dataset that was not exactly intentional: all the polygons in our dataset are regular, that is, for a given polygon, all the angles are equal and all the edges have the same length. This presents an interesting case of symmetry where:

- a combination of shapes from one class fit together nicely, without any gaps or overlaps to create another shape from another class. For example, 4 triangles fit perfectly in a square; and
- as we increase the number of edges, the polygons start to resemble to a circle, without ever becoming a circle.

We wonder how this symmetry effects the model and were curious to see if the model would learn better discriminative features if we trained using irregular polygons. While trying to generate irregular polygons, we realized that we could simulate a similar effect by using the eraser augmentation in the PyTorch framework. This augmentation randomly erases a section of the image, simulating the effect of disrupting the symmetry of polygons. Training the model with this added augmentation, improved test accuracy by ca. 10% (Table 1, Entry 2 vs 3).

A closer look at the confusion matrix from this model revealed the circle, heptagon, and octagon classes as problematic classifications (i.e., < 80% testing accuracy for each with other shapes within 10-15%). Circles were predominately misclassified as pentagons or nonagons, while heptagons and octagons were predominately misclassified as either pentagons or stars. Analysis of specific misclassified samples revealed that more than 90% of the samples had shapes ranging from 10-25% (smaller polygons) or 90% of the image width (larger polygons). This supports our hypothesis that the model will struggle to correctly classify shapes across a more diverse scale range than the model was trained on.

We also observe the model particularly struggles to classify smaller polygons. Intuitively, this makes sense, because we do not expect the distinguishing features represented by a learned filter for a smaller shape to translate to features when applied on a shape over different scales.

We can naturally circumvent this problem by training the model on polygons across diverse scale range, forcing the model to learn features that are robust to scale variance. We achieve this by applying zoom-in and zoom-out augmentations during training. This model misclassified only 34 shapes from our testing dataset, with heptagon contributing to 50% of misclassifications.

[2]Experiments performed by Ayyagari and Gillis.

$$\begin{bmatrix} -1 & -8 \\ 8 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -8 & 1 & 8 & -1 \\ -1 & 1 & 9 & 1 & -1 \\ -1 & 8 & 1 & -8 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -5 & 1 & 5 & 1 & -1 \\ -1 & -5 & -3 & 5 & 3 & 5 & -1 \\ -1 & -1 & 1 & 8 & 1 & -1 & -1 \\ -1 & 5 & 3 & 5 & -3 & -5 & -1 \\ -1 & 1 & 5 & 1 & -5 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

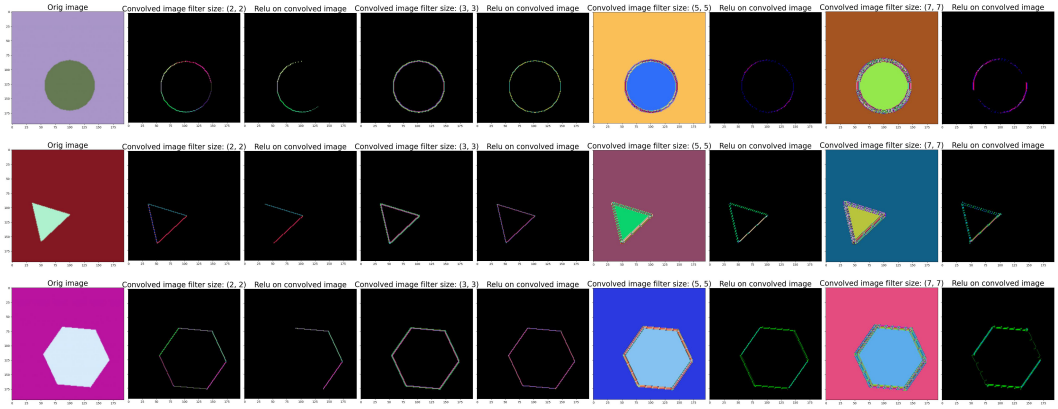


Figure 3: Filters used and visualization of convolved and ReLU activated images. Top: 2×2, 3×3, 5×5 and 7×7 edge-filters. Bottom left-to-right (edge-filter convolution or ReLU activation): Columns: 1, raw images; 2, 2×2 edge filter, 3, ReLU; 4, 3×3 edge filter, 5: ReLU; 6, 5×5 edge filter; 7, ReLU; 8, 7×7 edge filter, 9: ReLU.

5.2 Filter Sizes

5.2.1 Hypothesis

We envision a correlation between the filter sizes and the scales of the polygons the model learns to classify. Our intuition stems from our hypothesis that larger filter sizes, having larger receptive fields would encapsulate a stronger global definition of a shape, while the smaller filter sizes would encapsulate stronger local features that represents a shape.

To better understand the effects of filter sizes, we convolved images from our dataset with different sized edge filters (Figure 3) and observed that the 2×2 and 3×3 filters provided distinct continuous edges compared to larger sized filters. We proceeded to train different models varying the filter sizes of the first layer and expect to see better performance in models trained with smaller filter sizes.

5.2.2 Experimentation and Analysis

Table 1, entries 5–7 show the results for experiments performed by varying filter sizes in the first layer. As hypothesized, the model trained with 2×2 filter in the first layer results in the best performance on the test set. Most of the misclassifications for this model are small shapes from the classes circles, octagons, and nonagons.

We also note that as the filter sizes increase, the number of misclassifications for larger shapes increased. For the 7×7 filter we obtained several larger shapes that occupied most of the image; however, these were not present when using the 2×2 filter, and for the 5×5 filter it ranged between other two.

5.2.3 Multiple Sized Filters in First Layer

Motivated by the misclassification analysis from the filter size experiments we were curious to study the behavior of a model trained with all three filter sizes in the first layer of the neural network architecture. We expect that this model would be able to learn distinguishing features representing a combination of local and global features, resulting in a scale invariant neural network architecture.

We trained a neural network with filters 2×2, 5×5 and 7×7 in the first layer, concatenated the feature maps from these layers, and passed concatenated feature maps to the downstream layers. We also increased the number of dense layers to 3, and added batch normalization after every convolution layer to increase stability in learning. This model only misclassified 29 images, with about 51% of them being circles. We also note that all the misclassifications belong to the same size range as the shapes in the training set (Appendix, Figure A.1.19). We argue that this model is scale invariant.

5.3 Ensemble Network with Models Trained on Images with Different Resolutions

We were curious to see if we could also generate a scale invariant neural network by creating an ensemble of networks trained on different image resolutions: 64×64 and 128×128 and 200×200. We trained 2 additional models on resolutions 64×64 and 128×128 with some architectural modifications. For training the neural network on images with 64×64 resolution, we removed the pooling layers after the first 2 convolution layers and for the model trained on 128×128 images, we removed the pooling layer after the first convolution layer.

We combine the predictions from these models by computing the maximum of the predictions from each of these two models and the model in Table 1 Entry 5. This model misclassified 78 images from the test set with most of the misclassifications resulting from images with smaller shapes. We propose that better aggregation techniques and in-depth analysis into the misclassifications of this model to be the subject of future work.

Table 1: List of different experiments performed by varying parameters for filters and data augmentations techniques.

(a) Scale Invariance:				Accuracy (%)		
Entry[1]	Filters[2]	Augmentations[3]	Incorrect	Training	Validation	Testing
1[4]	a	none	574	91.8	45.0	36.2
2	a	a	192	98.5	98.0	78.7
3	a	a, b	160	99.0	99.8	82.2
4	a	a, b, c, d, e	34	95.6	99.8	96.2
5	b	a, b, c, d	90	99.3	99.9	90.0
6	c	a, b, c, d	123	95.8	99.4	86.3
7	d	a, b, c, d	145	97.4	99.9	83.8
8a	e	a, b, c, d	29	99.8	99.9	96.8
8b	f	a, b, c, d	78	99.2	99.8	91.3
(b) Translation Invariance:				Accuracy (%)		
9	a	none	726	100.0	94.7	19.3
10	a	a	678	94.6	97.7	24.7
11	a	d	244	98.1	98.4	72.9
12	a	b, d	174	96.2	99.6	80.7
13[5]	a	none	741	98.0	82.2	17.7
14[4]	a	b, d	167	95.8	99.4	81.4
15[5]	a	b, d	222	95.0	99.6	75.3

[1] Entry (all): Unless specified, network parameters were adopted from Figure 2.
[2] Filters: Filters: a, $\text{conv}_1(\text{filter}_1\text{: size=(3, 3)})$; b, $\text{conv}_1(\text{filter}_1\text{: size=(2, 2)})$; c, $\text{conv}_1(\text{filter}_1\text{: size=(5, 5)})$; d, $\text{conv}_1(\text{filter}_1\text{: size=(7, 7)})$; e, $\text{conv}_1(\text{filter}_1\text{: concat(size=(7, 7);size=(5, 5);size=(2, 2))}$ f, 64 x 64: a;128 x 128, a;200 x 200, b.
[3] Augmentations: none; a, rotation; b, eraser; c, flips; d, affine; e, zoom-in & zoom-out.
[4] Dense layers: Entry 1, 1 dense layer; entry 14, 3 dense layers.
[5] No maxpool layers for layers 2 and 3.

6 Towards Translation Invariance[3]

For this study, we wanted to understand how our convolutional model learn to be invariant to translations (i.e., the spatial location of a shape). We performed experiments with training and validation data with shapes at the center of the image and for the testing data, shapes were proximal to the edges of the image. Without data augmentation, the testing accuracy that we achieved was 19.3% which was what we expected.

6.1 Data Augmentation

As reported by Biscione et al. [3] and Kauderer-Adams [4], data augmentation is the primary cause for translation invariance. We experimented with data augmentation in order to provide our training model with images where the position of the shapes are translated and dispersed around the image. Our hypothesis was that it will force our model to reduce its dependency on positional features and it will try to learn essential features for classification.

1. For the first augmentation experiment, we applied random rotation [5] on the images for training data. The model did not perform well on testing data - it only achieved 24.7% accuracy (Table 1, Entry 10). One of the interesting results we obtained was our model classified more than 60% of the testing images under the star category when in reality the stars were only 10% of the testing data (Appendix, Figure A.2.5).

2. For the second augmentation experiment, we trained our model with affine augmentation [5]. By using this technique, we were able to obtain training dataset with shapes dispersed in the image as we can be observed in Appendex, Figure A.2.2. We tried different degrees for translation and discovered that a maximum of 20% was the best parameter without getting the shapes out of the image. The model obtained 98.1% accuracy in training and 98.5% in validation accuracy. The testing accuracy was 72.9% (Table 1, Entry 11). For shapes like triangle, pentagon, and square the model has overly classified (Appendix, Figure A.2.7).

3. The best combination of augmentation was with affine and random erasing [5]. We assumed that by randomly erasing some portions of an image, the model will reduce dependency on general features and it will try to learn more detailed features of the shape. The model achieved 80.7% testing accuracy with these augmentations (Table 1, Entry 12; Appendix, Figures A.2.8 and A.2.10).

6.2 Architectural Experiments

By applying maxpool we loose important information during training [6]. Since maxpool creates information loss, we hypothesize that performing some architectural changes like removing the maxpool layer and adding dense layers, we can achieve higher translation invariance for our model. Thus, we experimented with removing maxpool layers from the second and third convolutional layers and added a dense layer with and without the augmentation (Table 1, Entries 13–15). We observed that by removing maxpool layers from second and third layer did not helped us in terms of accuracy gain, but the category wide classifications were more balanced except hexagon (Figure A.2.15. From this experiment we observed that removing maxpool layer decreased the overall accuracy; however, the classification was more balanced (Table 1, Entries 12 and 15).

For the architectural experiments, adding a dense layer provided the highest accuracy for our testing data. The most under-classified shape was circle which was mostly misclassified as higher faceted shapes like octagons and nonagons (Appendix, Figure A.2.14). Moreover, we observed that the images that were misclassified were located on the edges of the images. We observed the same pattern for misclassification for architectural experiment where 2 maxpool layers were removed (Appendix, Figure A.2.14).

[3]Experiments performed by Lakhani and Pramir.

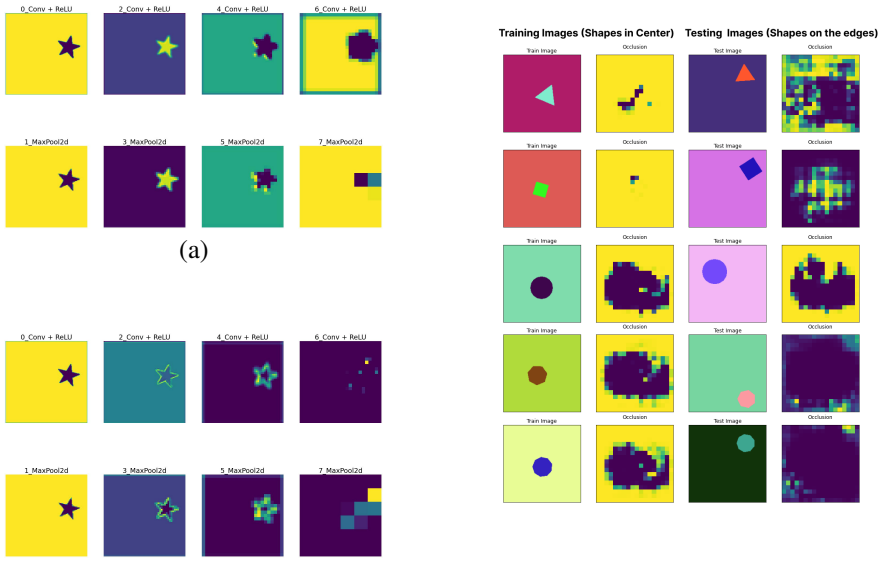


Figure 4: Maximally activated feature maps for convolution and pooling layers for a poor (a) and good (b) trained models. (c) Heatmaps for the occlusion experiments for Entry 12 in Table 1.

7 Visualization

Although misclassification analysis was giving us some insights about the decisions of a model, we explored two visualization techniques to further our understanding and interpretation of the decisions made by the model.

Our first approach was to look at the model weights. We visualized the maximally activated feature maps from the convolution and pooling layers. We observed that the initial convolution layers learned human readable shapes while the last layer learned more abstract representations of a shape. When we contrasted the visualizations from a poorly performing model with a better performing model across different epochs during training, we noticed that the better performing model presented more varied features over training while the poorer performing model shows very subtle changes in feature maps during the training of the model.

Our second approach was to perform an occlusion experiment as described by Kumar [7] which provided heatmaps to represent spatial locations of the image that are predominately responsible for classification. We expected a well-trained model to localize the shapes of the polygons as the most relevant regions for the model's classification decision. We conducted experiments by passing images from training and testing sets through our best performing models and visualized the resulting heatmaps (Table 1, Entry 12). We observed that for images in the training set, the heatmaps localized spatial locations corresponding to the shapes as expected, but for the images in the test set, the heatmaps identified regions spatially dispersed across the image without much correlation with the location of the shape. (Appendix, Figure **??**). We hypothesize that the network is confused by these test images and is arriving at the classification decisions based on associations that we are unable to interpret.

8 Conclusion

We explored network parameters like type of pooling, filter sizes and training strategies like augmentation to successfully design and train convolution neural networks for both the scale and transitionally varied synthesized datasets. We start with a simple convolution neural network; analyse misclassifications; carefully propose incremental changes; hypothesise, execute, validate and analyse the proposed changes to design these neural networks. We choose a limited number of network parameters and augmentation strategies to explore, and attempt to thoroughly understand the influence of these parameters and training strategies in the context of the dataset designed and the problem they are being

applied to. We attempt to explain our models' decisions by visualising maximally activated feature maps and performing an occlusion experiment. We deduce that more sophisticated visualizations strategies are required to explain the models' decisions.

Through the choices we make in designing these models we highlight the importance of paying attention to both the design choices of a network architecture and carefully inspecting the misclassifications while training neural networks.

9 Acknowledgements

We would like to thank Sageev Oore and Marvin da Silva for the helpful comments and suggestions.

References

[1] A. E. L. Korchi. "2D geometric shapes generator." (2022), [Online]. Available: https://github.com/elkorchi/2DGeometricShapesGenerator (Accessed: April 5, 2022).

[2] TensorFlow. "Basic classification: Classify images of clothing." (2022), [Online]. Available: https://www.tensorflow.org/tutorials/keras/classification (Accessed: April 4, 2022).

[3] V. Biscione and J. Bowers. "Learning translation invariance in CNN." arXiv: 2011.11757v1 [cs.CV]. (2020), [Online]. Available: https://arxiv.org/abs/2011.11757 (Accessed: April 10, 2022).

[4] E. Kauderer-Adams. "Quantifying translation-invariance in convolutional neural networks." arXiv: 1801.01450v1 [cs.LG]. (2017), [Online]. Available: https://arxiv.org/abs/1801.01450 (Accessed: April 10, 2022).

[5] PyTorch. "Transforming and augmenting images." (2022), [Online]. Available: https://pytorch.org/vision/stable/transforms.html (Accessed: April 10, 2022).

[6] H. Su, F. Liu, Y. Xie, and F. Xing. "Region segmentation in histopathological breast cancer images using deep convolutional neural network." (2015), (Accessed: April 10, 2022).

[7] N. Kumar. "Deeplearning-padhai." (2022), [Online]. Available: https://github.com/Niranjankumar-c/DeepLearning-PadhAI (Accessed: April 7, 2022).

A Appendix

A.1 Scale Invariance

A.1.1 9 Classes of Geometric Shapes

Figure A.1.1: Geometric shapes with image dimensions of 200×200. Top panel, training/validation; bottom panel, testing.

A.1.2 Confusion Matricies and Misclassifications

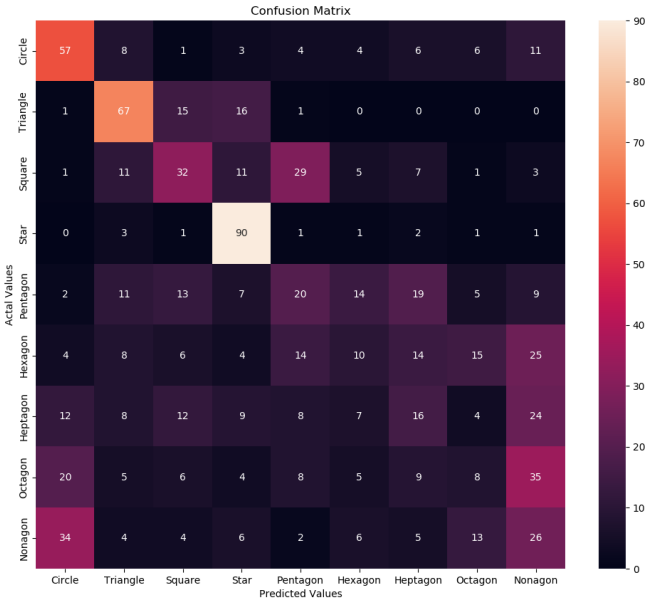


Figure A.1.2: Confusion matrix for Entry 1 from Table 1.

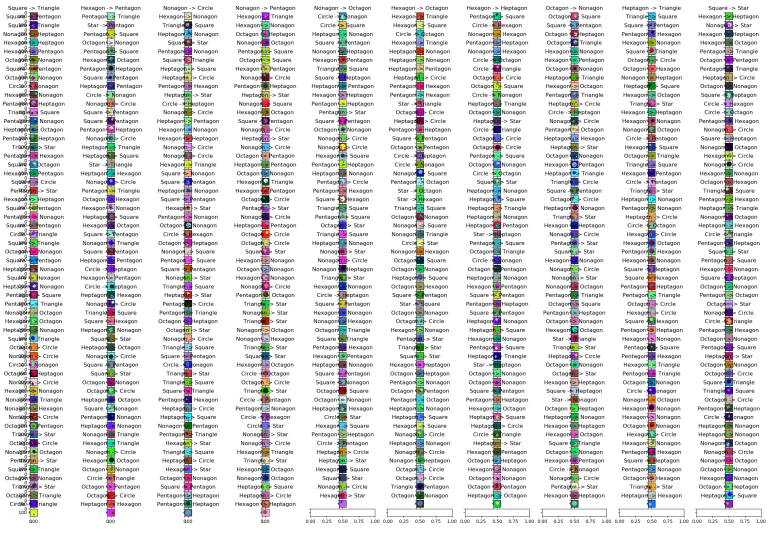


Figure A.1.3: Misclassifications for Entry 1 from Table 1.

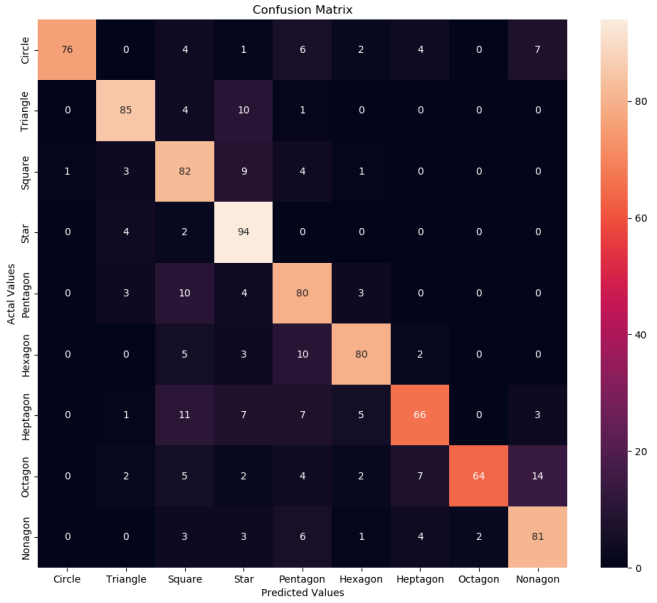


Figure A.1.4: Confusion matrix for Entry 2 from Table 1.

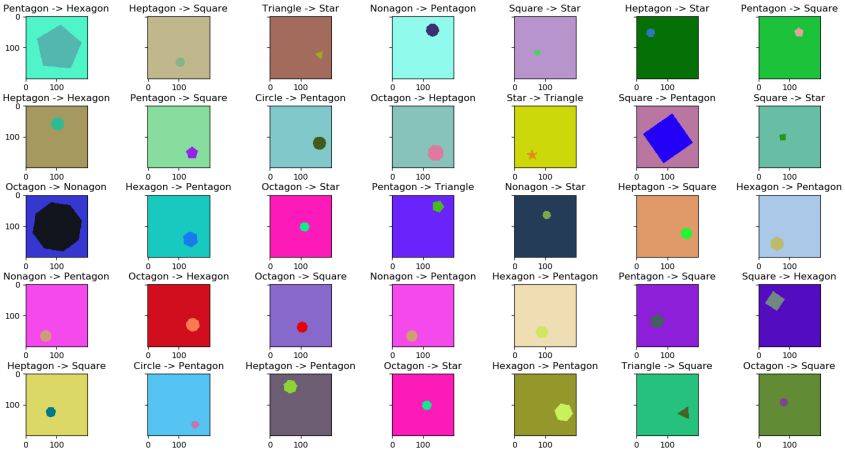


Figure A.1.5: Misclassifications for Entry 2 from Table 1.

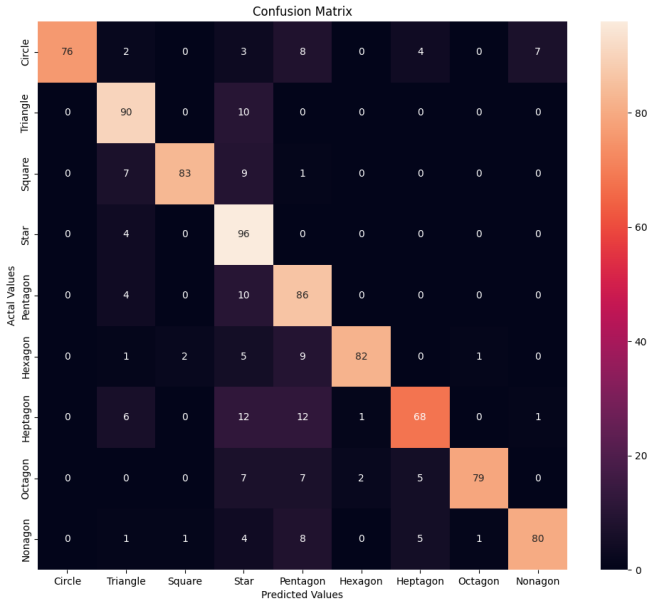


Figure A.1.6: Confusion matrix for Entry 3 from Table 1.

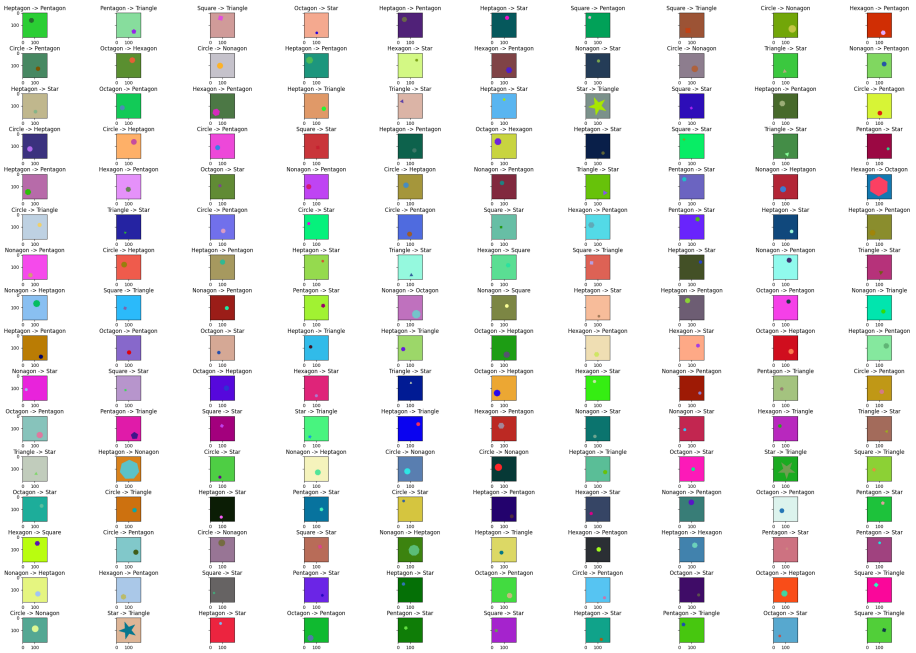


Figure A.1.7: Misclassifications for Entry 3 from Table 1.

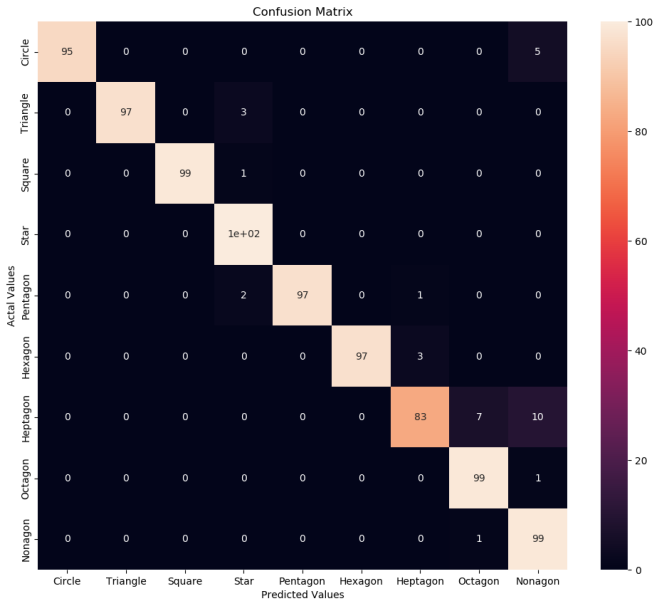


Figure A.1.8: Confusion matrix for Entry 4 from Table 1.

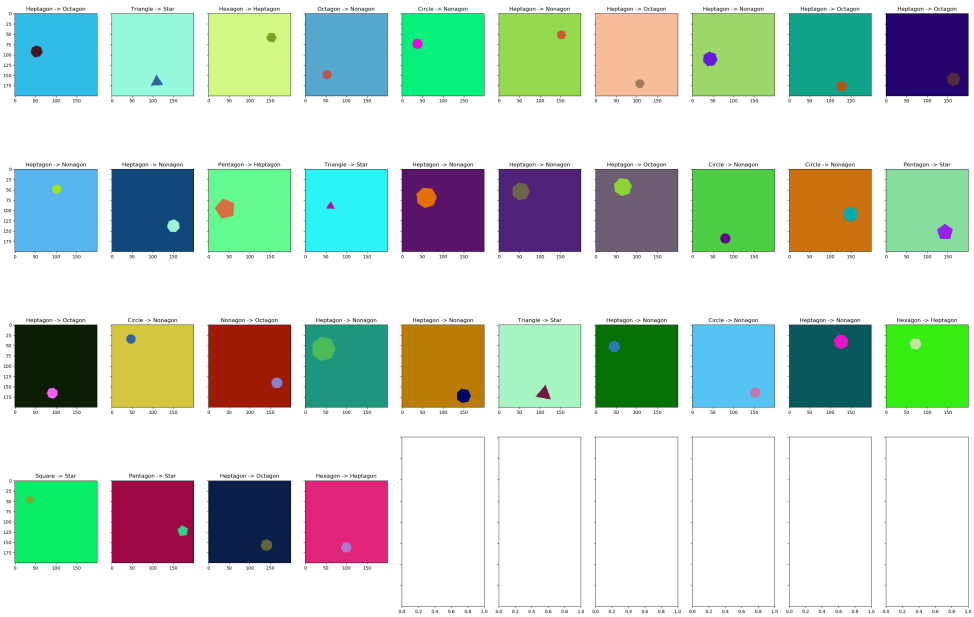


Figure A.1.9: Misclassifications for Entry 4 from Table 1.

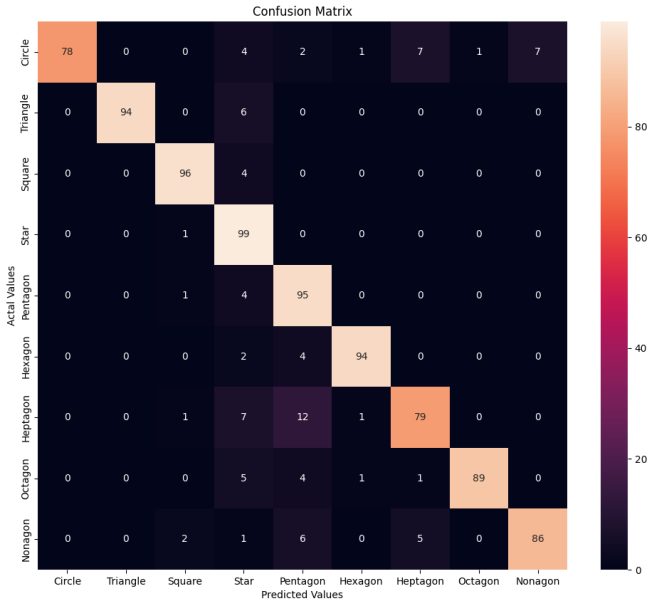


Figure A.1.10: Confusion matrix for Entry 5 from Table 1.

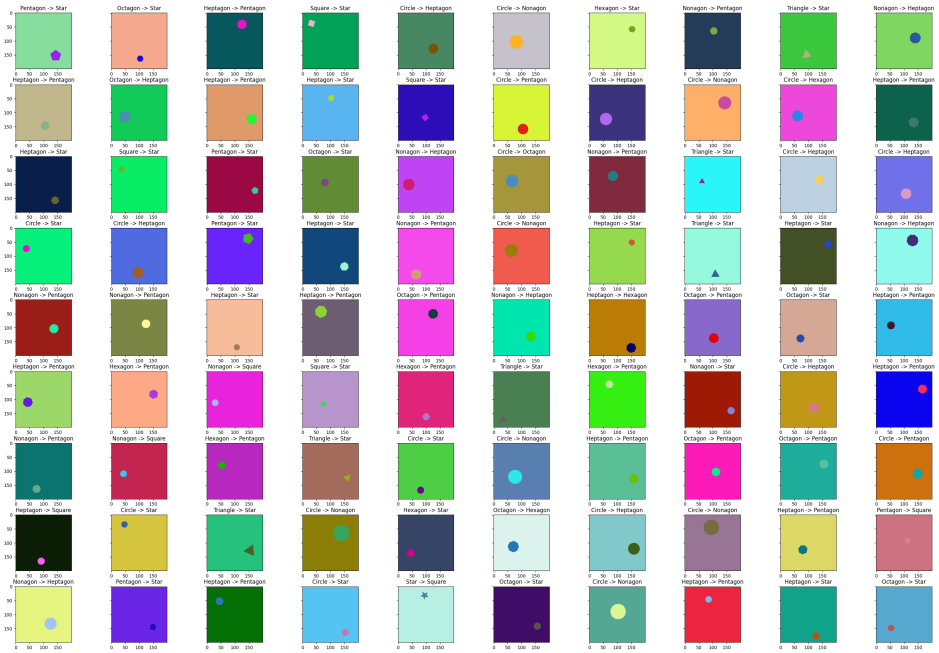


Figure A.1.11: Misclassifications for Entry 5 from Table 1.

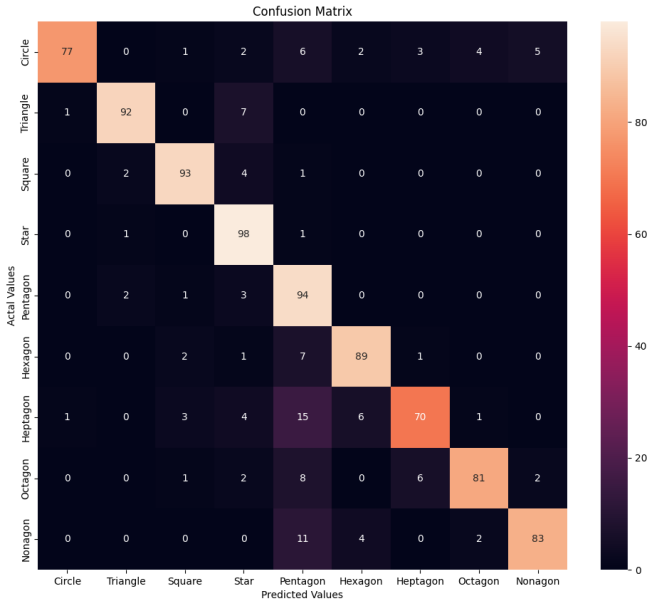


Figure A.1.12: Confusion matrix for Entry 6 from Table 1.

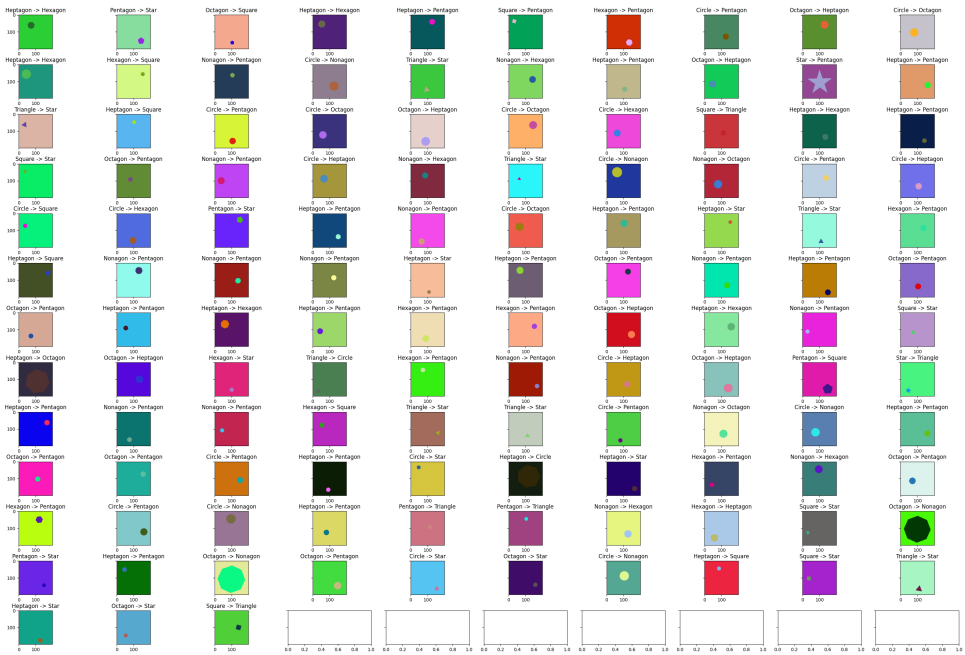


Figure A.1.13: Misclassifications for Entry 6 from Table 1.

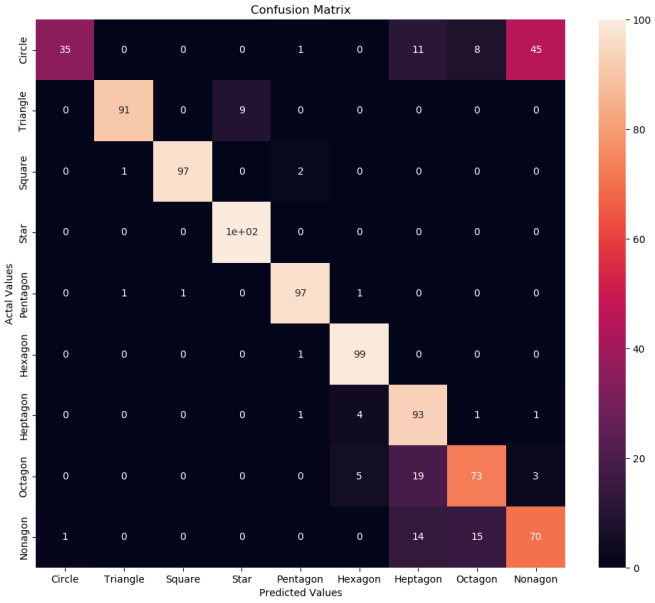


Figure A.1.14: Confusion matrix for Entry 7 from Table 1.

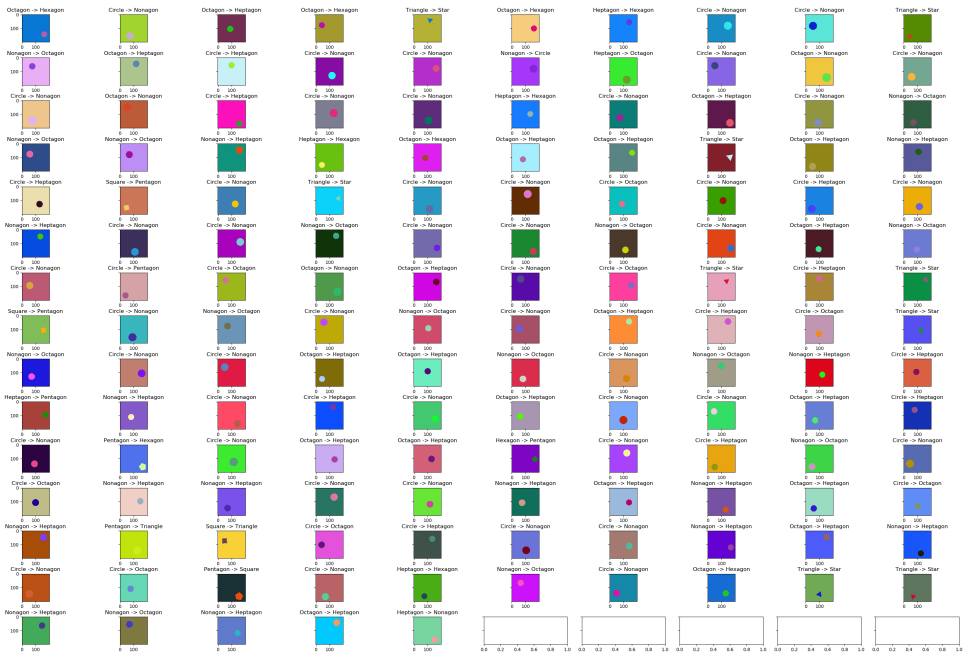


Figure A.1.15: Misclassifications for Entry 7 from Table 1.

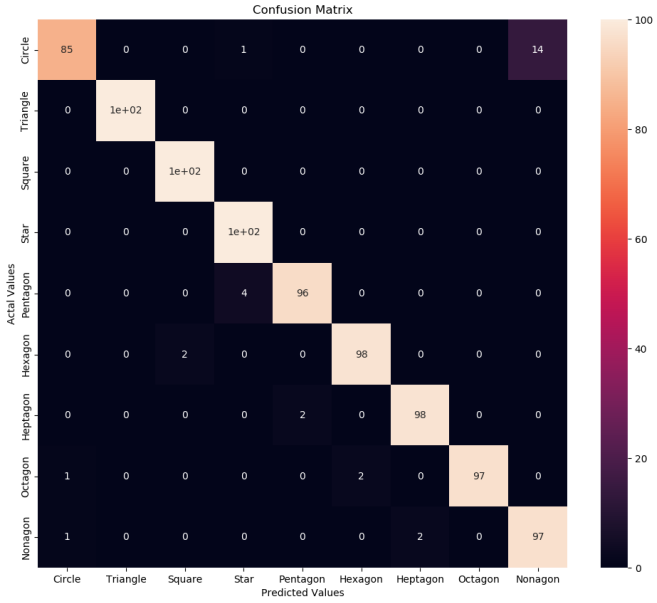


Figure A.1.16: Confusion matrix for Entry 8a from Table 1.

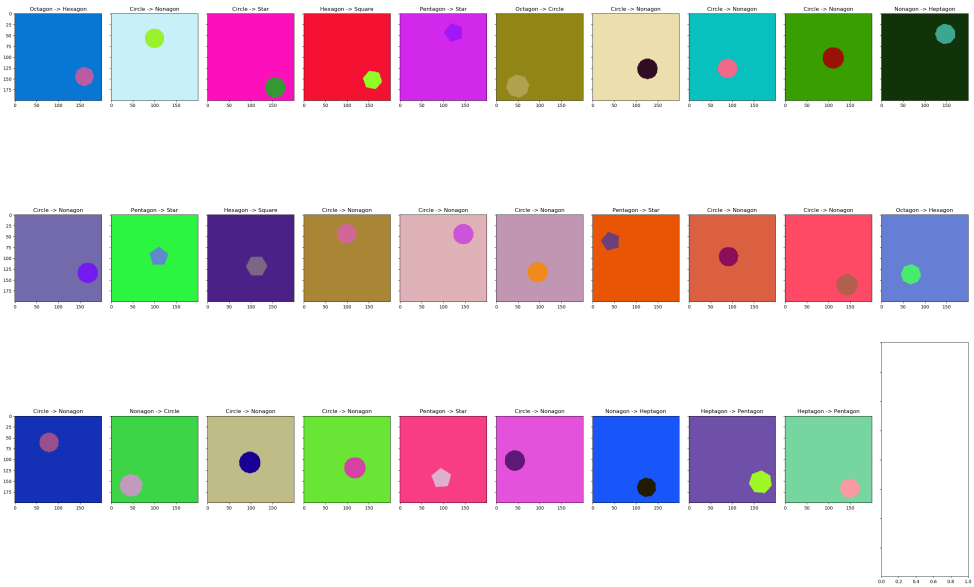


Figure A.1.17: Misclassifications for Entry 8a from Table 1.

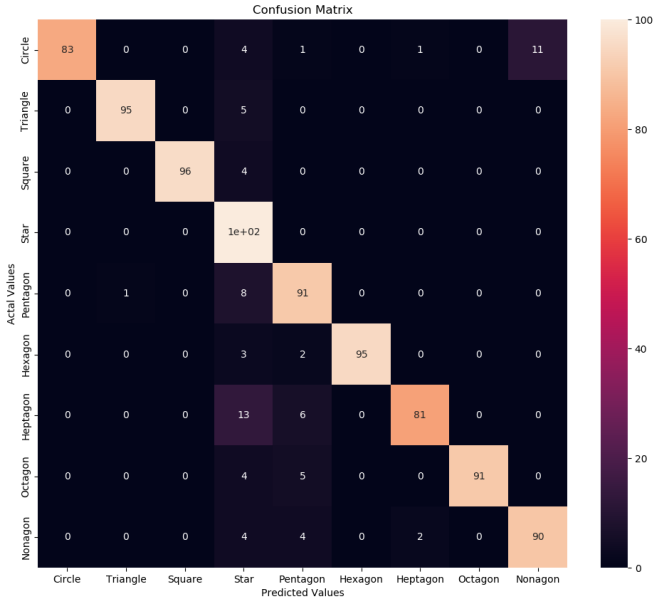


Figure A.1.18: Confusion matrix for Entry 8b from Table 1.

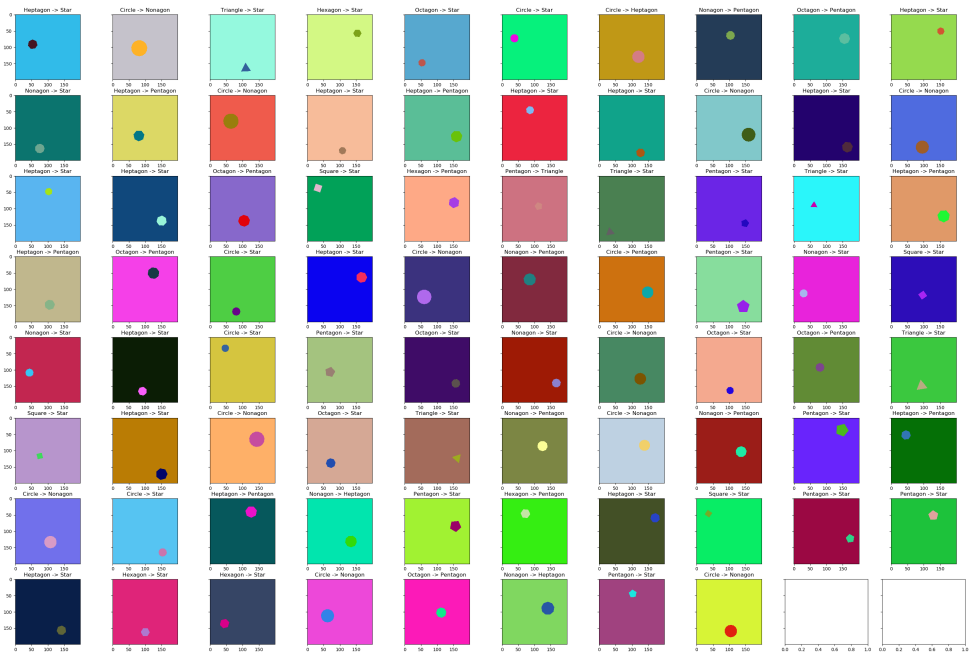


Figure A.1.19: Misclassifications for Entry 8b from Table 1.

A.2 Translation Invariance

A.2.1 9 Classes of Geometric Shapes

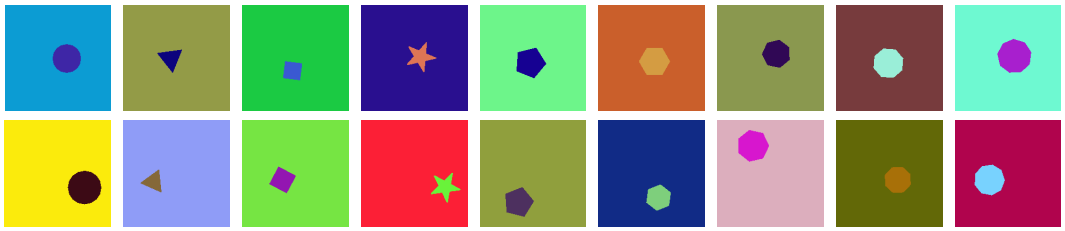


Figure A.2.1: Geometric shapes with image dimensions of 200×200. Top panel, training/validation; bottom panel, testing.

A.2.2 Augmentations

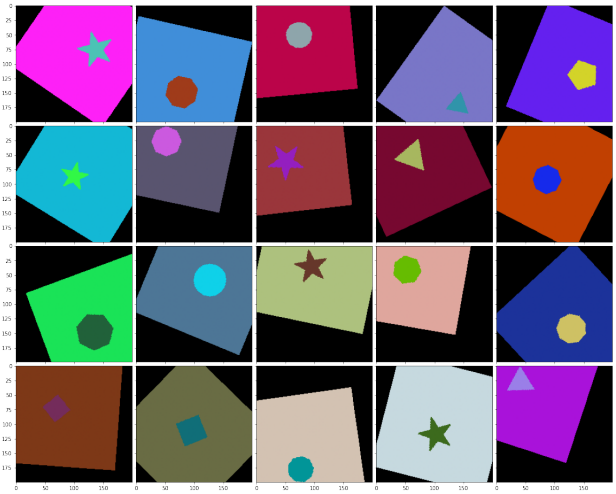


Figure A.2.2: Training images after affine transformation.

A.2.3 Confusion Matricies and Misclassifications

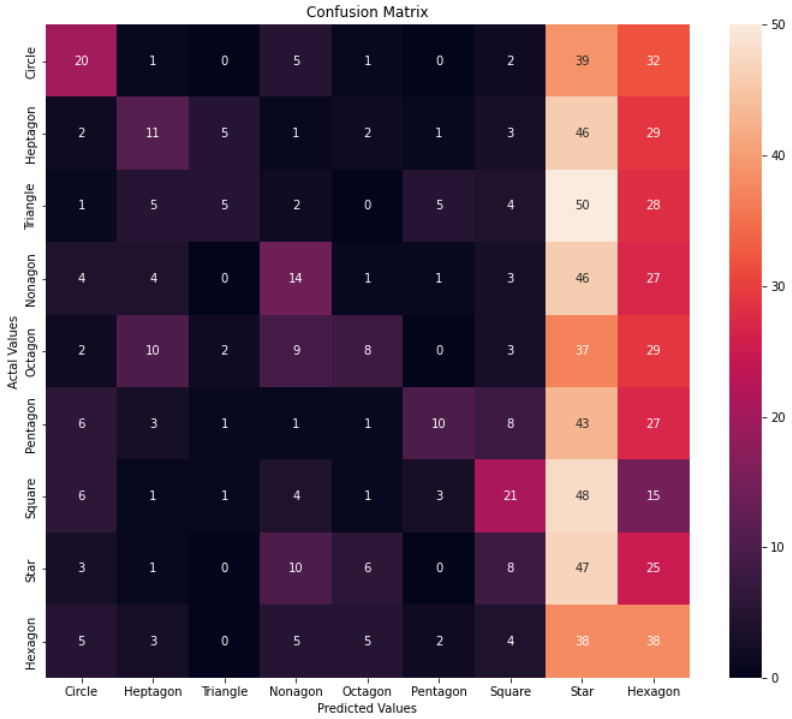


Figure A.2.3: Confusion matrix for Entry 9 from Table 1.

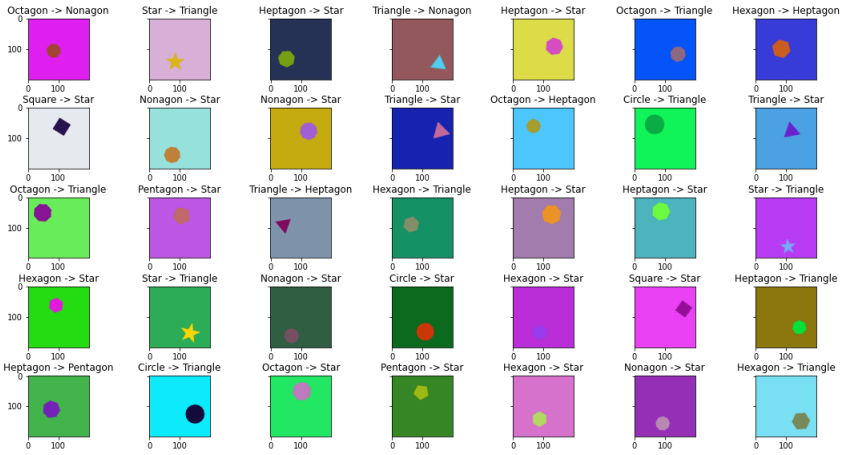


Figure A.2.4: Misclassifications for Entry 9 from Table 1.

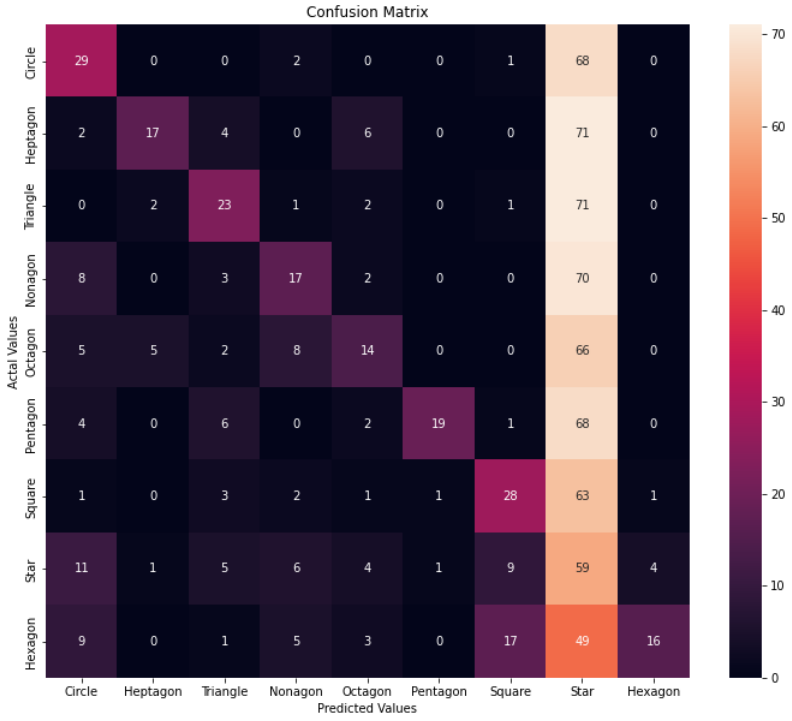


Figure A.2.5: Confusion matrix for Entry 10 from Table 1.

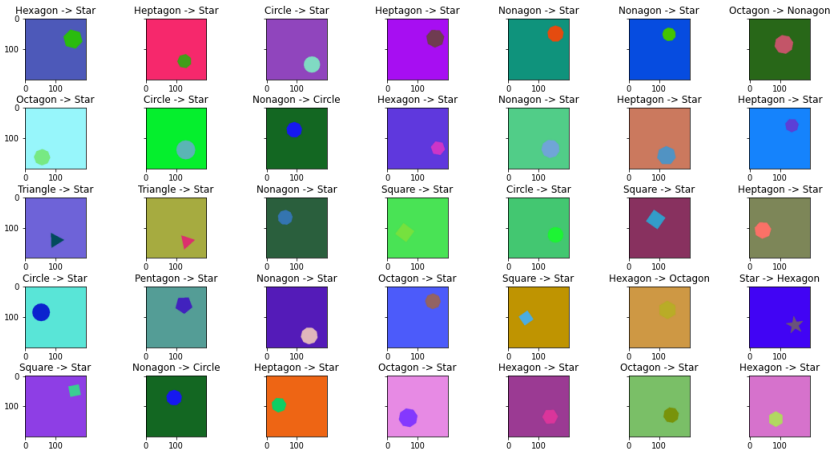


Figure A.2.6: Misclassifications for Entry 10 from Table 1.

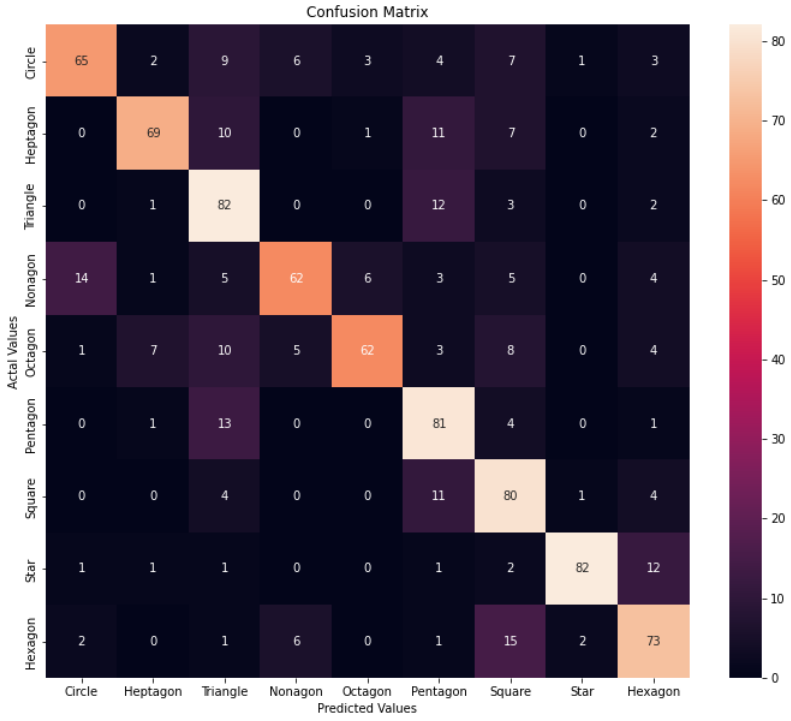


Figure A.2.7: Confusion matrix for Entry 11 from Table 1.

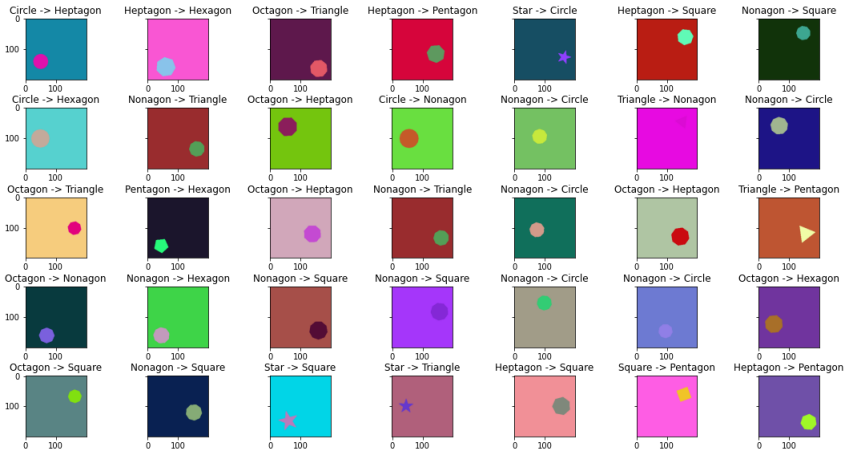


Figure A.2.8: Misclassifications for Entry 11 from Table 1.

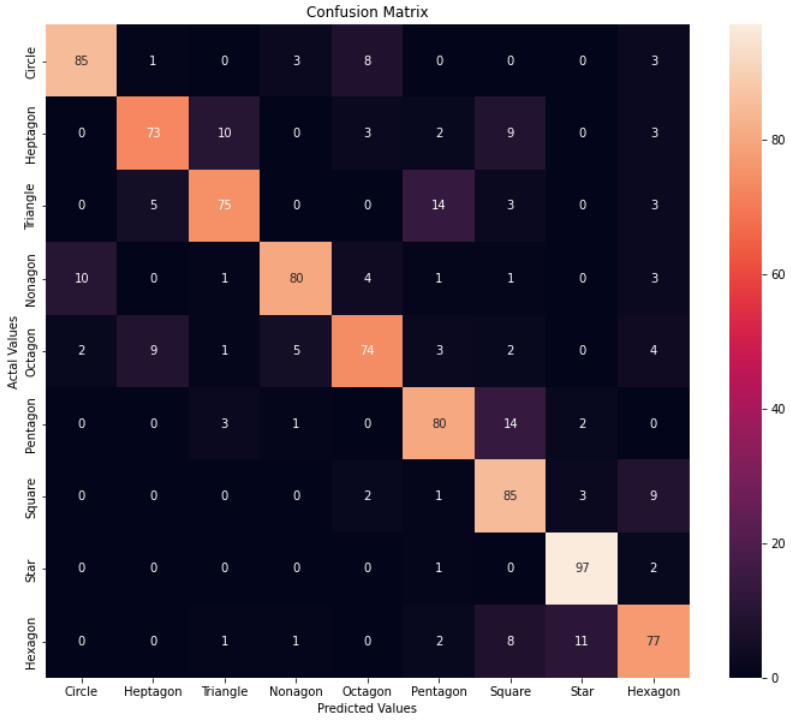


Figure A.2.9: Confusion matrix for Entry 12 from Table 1.

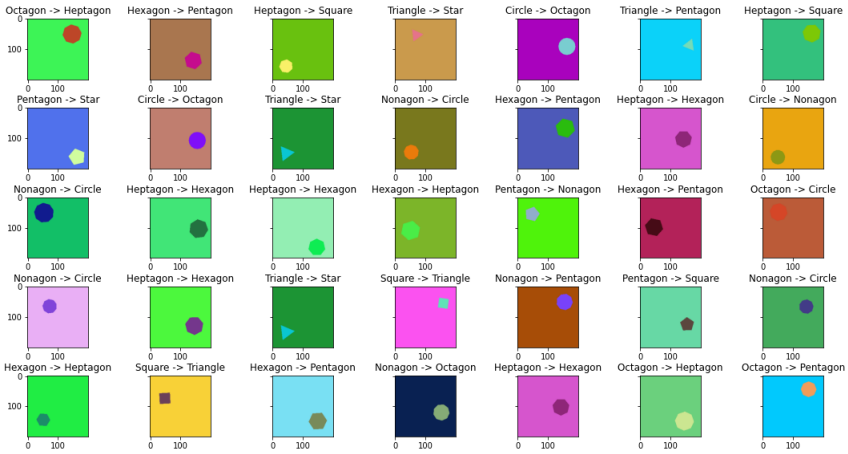


Figure A.2.10: Misclassifications for Entry 12 from Table 1.

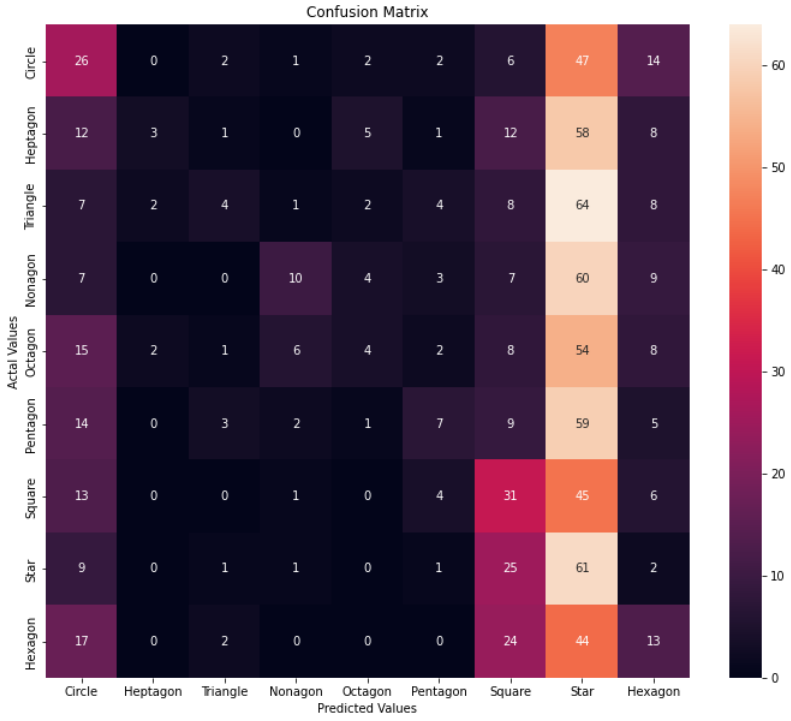


Figure A.2.11: Confusion matrix for Entry 13 from Table 1.

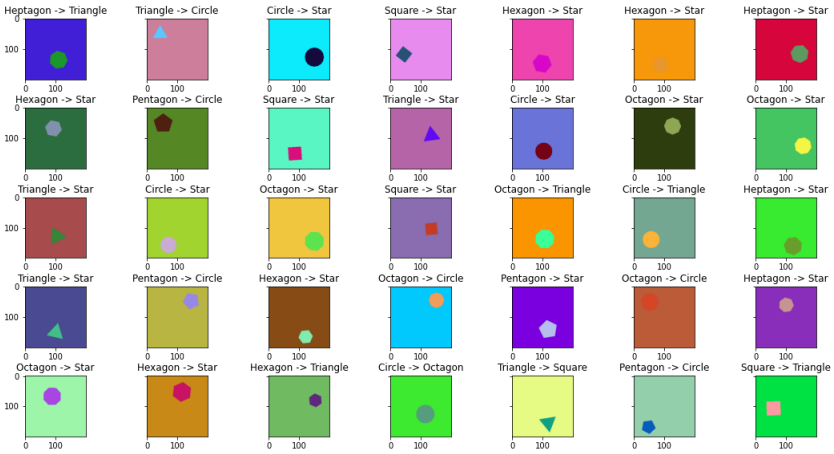


Figure A.2.12: Misclassifications for Entry 13 from Table 1.

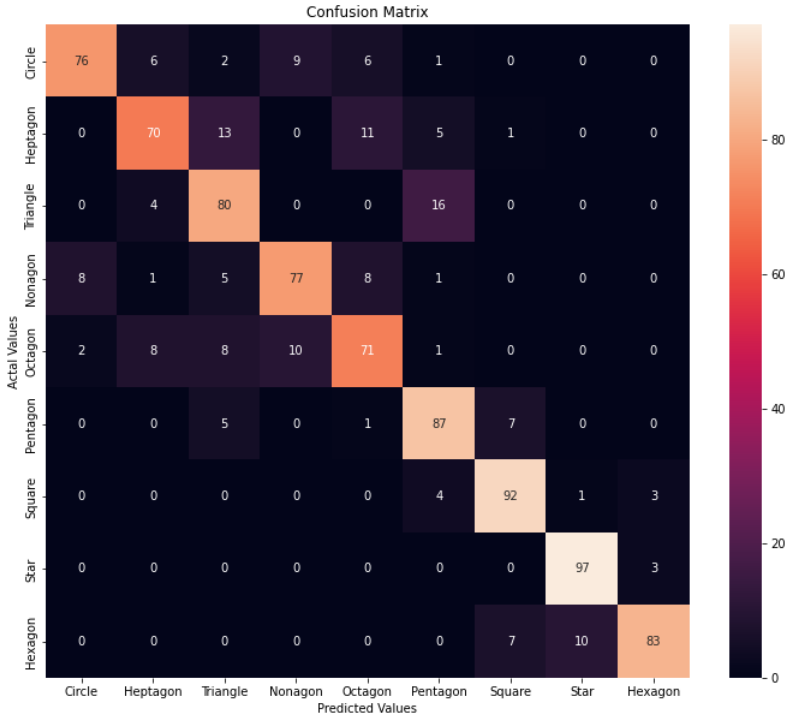


Figure A.2.13: Confusion matrix for Entry 14 from Table 1.

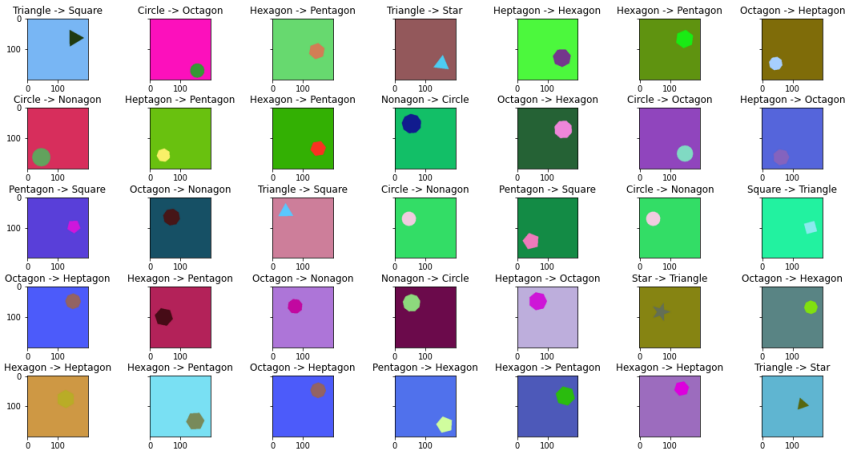


Figure A.2.14: Misclassifications for Entry 14 from Table 1.

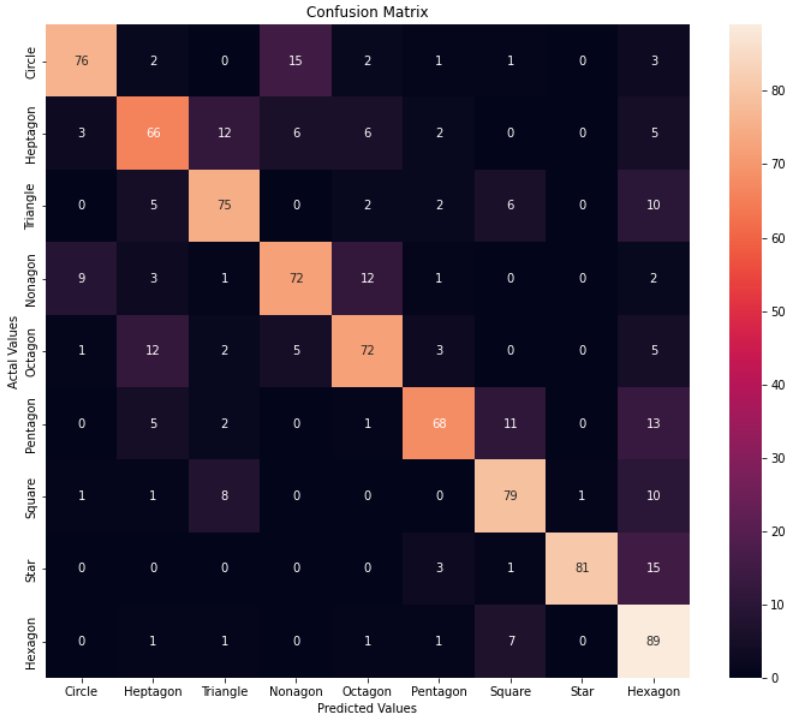


Figure A.2.15: Confusion matrix for Entry 15 from Table 1.

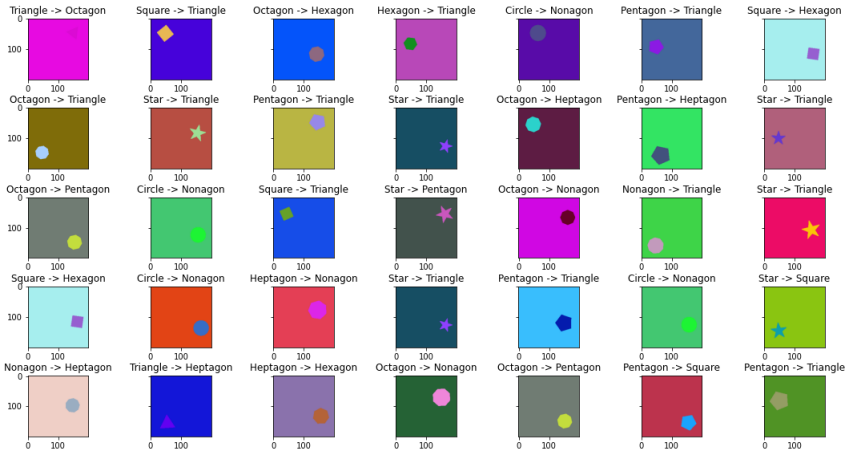


Figure A.2.16: Misclassifications for Entry 15 from Table 1.