Mixed and Pure Breed Dog Classifier

Luke Trinity and Nat Shenton
CS 254

December 10, 2019

1 Introduction

Online sources of high resolution images including Google, Image-net [1], and Flickr are providing new opportunities to explore applications of image categorization using computer vision. One interesting application in the field of fine-grained image classification is identification of dog breed. Dogs faces are considered to be highly differentiable for each breed, which makes classification an approachable problem. However, the work is challenging due to the varying combinations of facial characteristics and color patterns between species, as well as intra-class variation within a single species. Another aspect of the problem that makes it more complicated is the presence of humans and man-made backgrounds, which are not as common in other animal datasets. In addition, many attempts to classify dog breeds fail to corroborate the results with actual DNA evidence. Our approach will be to utilize an existing dog breed image classification dataset [2], in combination with a novel dataset that includes paired DNA test results and photos [3] and crowd sourced mixed breed dog images, to classify dogs as pure bred or mixed breed.

2 Problem Definition and Algorithm

2.1 Task Definition

The task is to take dog images where each image is labeled either mixed or pure breed and use them to train an algorithm to classify an unseen photo based on whether the dog is mixed or pure. An important part of training our neural network is to normalize the input images. Each input image is defined as a matrix of $250 \times 250 \times 3$. The output of this algorithm will either be a mixed or pure bred dog. Potential benefits of this work are to provide dog owners an easy way to classify if their dog is a pure bred or mixed, or assist shelters in generating interest in dogs to increase adoption rates. From a machine learning perspective, our testing suite produces interesting and thought provoking results about how to select a model in a difficult decision space. It could be an exciting way to start a conversation and help people learn about different breeds of dogs, and how they are mixed. The final goal for this task would allow for a dog owner to upload a picture of their dog to a website or app and then the machine learning algorithm will classify their dog as the pure bred breed or the top two possible breeds.

2.2 Dataset

The Stanford Dogs Dataset has 20,580 images each classified into one of 120 classes of dog breed. [2] Each image also has an associated bounding box identifying the location of the dog within the photo. The mixed-breed DNA identified dog photo dataset from Voith et al. [3] has 20 images, each paired with DNA percentage breakdown of dog breed. The final mixed breed data used crowdsourcing to bring the total up to 266 pictures. There was no way to actually identify if the dog was actually mixed or not so we are assuming that the submission of the mixed breed dog is correct in the data. The final dataset was obtained by a random sampling of pure bred dogs from the Standard dataset equal to the length of the amount of dogs form the Stanford Dogs Dataset, giving us a balanced dataset of 532 images.

2.3 Algorithm Definition

First, we utilize the Resnet50 convolutional neural network [4] as it achieves roughly 80% accuracy on the Standford dog dataset in classifying pure bred dogs. This pretrained neural network gives a classification accuracy high enough that it should classify an image of a purebred dog correctly. The convolutional neural network is trained on Google's ImageNet which contains the Stanford Dogs dataset. By feeding images from the mixed dog breed dataset into this pretrained neural network, the network gives the output as the probability of each dog breed. The output from Resnet50 contains the probabilities of 1000 labels, and our algorithm used the top 4 highest probabilities. From here principal component analysis is used to take the four prediction outputs from the pretrained convolutional neural network down to two features. With these two features as the input classes, a sweep of four classification methods were used: linear kernel SVM, optimal kernel SVM, random forest, and a neural network.

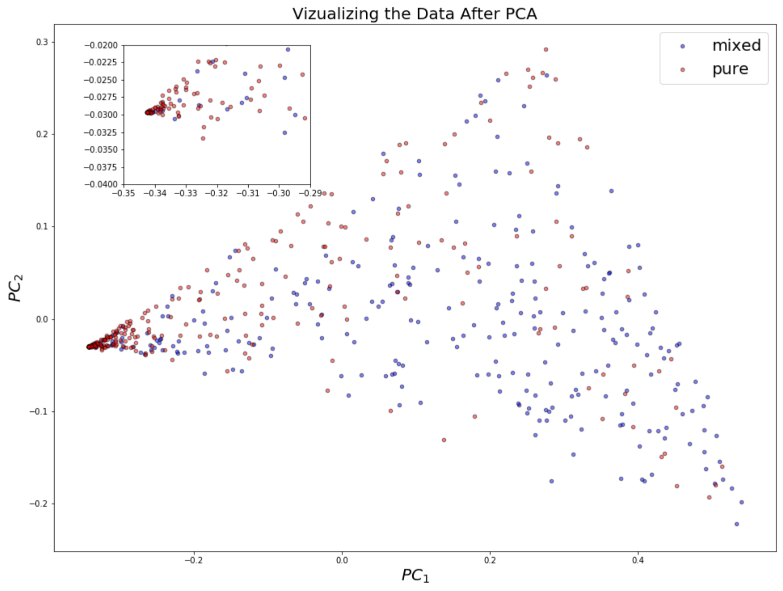

Figure 1: The distribution of the data after using principal component analysis to reduce the breed probabilities from 4 features to 2. Red is purebred while blue is mixed. The inset plot shows the area where the data in clustered.

3 Experimental Evaluation

3.1 Methodology

For the SVM linear kernel we used cross validation to search over C values randomly drawn from an exponential distribution (scale=1000), γ values randomized over an exponential distribution (scale = 2), and randomly choosing between balanced or no class weight. For the SVM optimal kernel, the cross validation search over Radial Basis Function kernel and polynomial kernel. The random forest algorithm looks at a forest of 30 trees, and draws randomly from a uniform distribution for the minimum number of samples to split on as fraction of the total number of samples. For random forest we also randomly validated for a max depth as a random integer between 1 and 50. Before moving on to neural network testing suite description, please note that for the cross validation specified above, 15 fold was used across model types, and the randomized search cross validation searched over 200 iterations for each model.

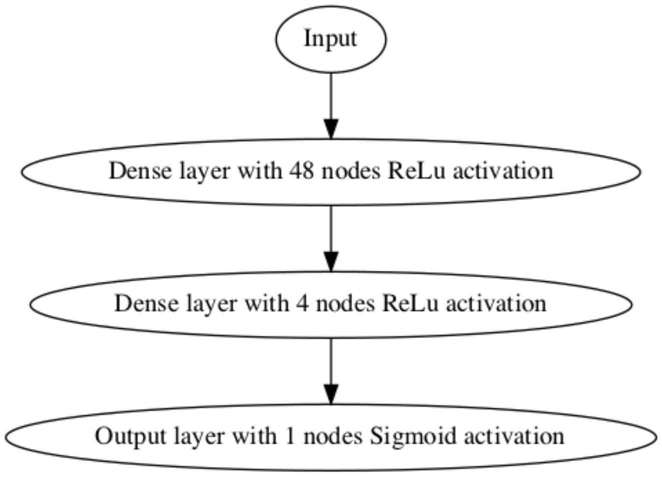
Input
Dense layer with 48 nodes ReLu activation
Dense layer with 4 nodes ReLu activation
Output layer with 1 nodes Sigmoid activation

Figure 2: Neural Network Structure

The neural network uses an architecture of 3 hidden fully connected layers with the ReLu activation function. The first layer contains 48 nodes, the second 24 nodes, and the third 4 nodes; each with bias and kernel constraints. The output layer contains one node with a sigmoid activation function. The model was then compiled with the Adam gradient descent optimizer, and binary cross-entropy as the loss function. The neural network was trained over 300 epochs. For each of the four model types described, we trained as described above, and tested 250 different splits of the data with a 66-33 training testing split. Here we report the accuracy, as well as the different between the training and test accuracy, for each of those 1000 models grouped by model type.

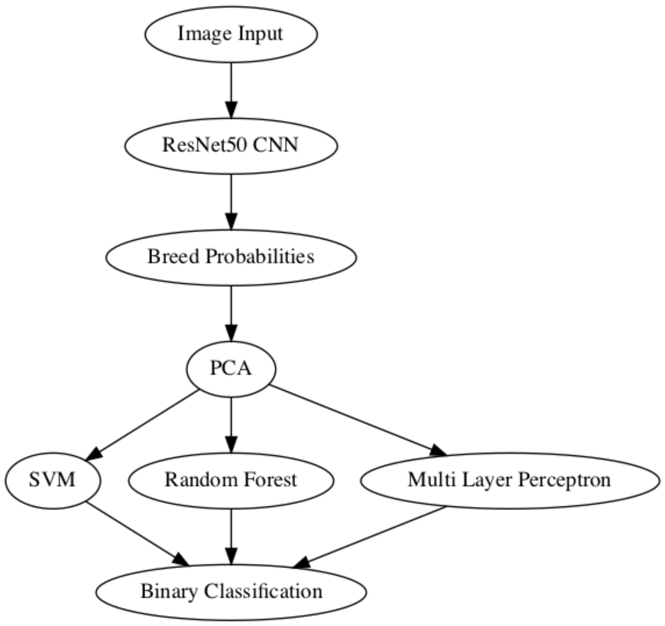

Figure 3: Model Testing Suite Visualization

3.2 Additional Neural Network Methodology

In the results section we present the conclusion that neural networks are performing the best on our dataset. To further test how different neural network architectures might affect accuracy, we increased and decreased the number of nodes in our hidden layers and ran a similar experiment. This second testing suite is used to test the three neural network architectures to increase performance. In addition to our initial neural network (48, 24, 4, and 1 nodes in each hidden layer) which we will refer to as 'medium' depth, we also test a 'less deep' model (24, 12, 2, and 1 nodes in each hidden layer) and a deeper model (96, 48, 8, 1 nodes in each hidden layer). The same bias and kernel constraints were used for each of the neural networks, which were trained for 300 epochs each on the same 250 splits on the data. For these 750 total models tested, we again report the accuracy and training-test accuracy distributions.

3.3 Results

Here are the results of the initial testing suite. Figure 4 shows the distribution of accuracy scores. Here this shows the initial testing suite results with each model run 250 times. The SVM are using random search cross validation what the only parameters changed in the testing.

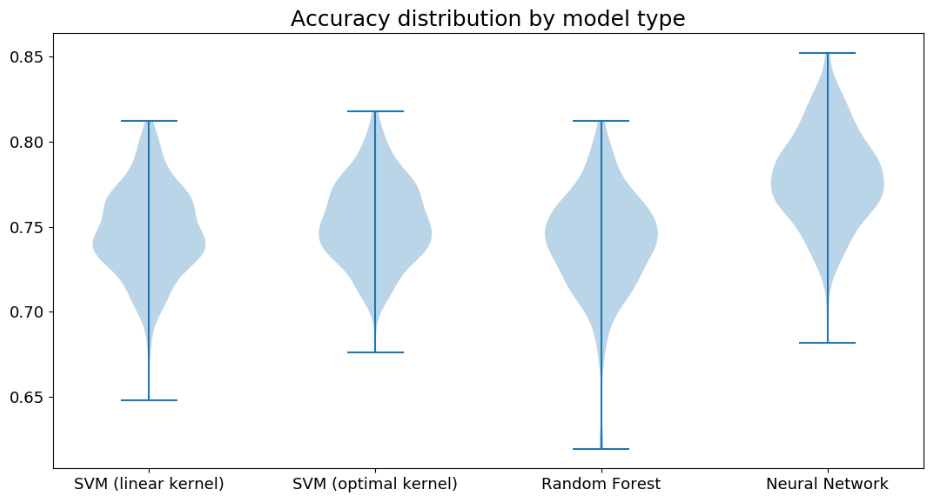

Figure 4: The violin plot shows the distribution of testing accuracy of each model type from the testing sweep.

Figure 4 shows that SVM with linear kernel, SVM with a non-linear kernel, and random forests perform about the same, but the neural network with the architecture shown in figure 1 gives the best accuracy. The table gives the numeric results of this suite.

Model Type	Mean	Standard Deviation
SVM (linear kernel)	0.747	0.026
SVM (optimal kernel)	0.753	0.024
Random Forest	0.742	0.030
Multi Layer Perceptron	0.780	0.028

Figure 5 gives the $\text{Accuracy}_{\text{Testing}} - \text{Accuracy}_{\text{Training}}$ for each model. This gives a representation of the overfitting and under-fitting the models could be experiencing. Here it seems that the neural network is overfitting the data with the mostly negative values.

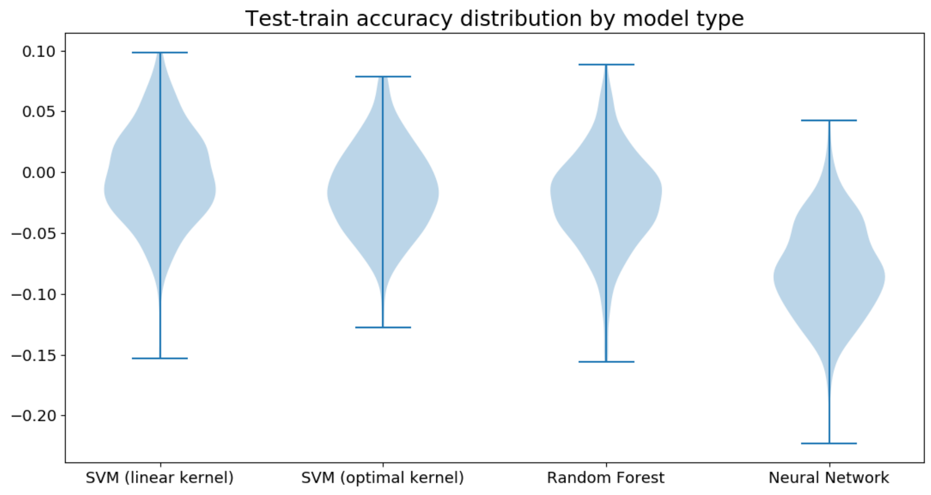

Figure 5: The violin plot shows the distribution of the $\text{Accuracy}_{\text{Testing}} - \text{Accuracy}_{\text{Training}}$ for each model.

The numeric results of the different between the testing and training is found in the table below.

Model Type	Mean	Standard Deviation
SVM (linear kernel)	-0.003	0.039
SVM (optimal kernel)	-0.014	0.036
Random Forest	-0.022	0.039
Multi Layer Perceptron	-0.080	0.039

The decision boundary of the used in first testing suite neural network is found in figure 6.

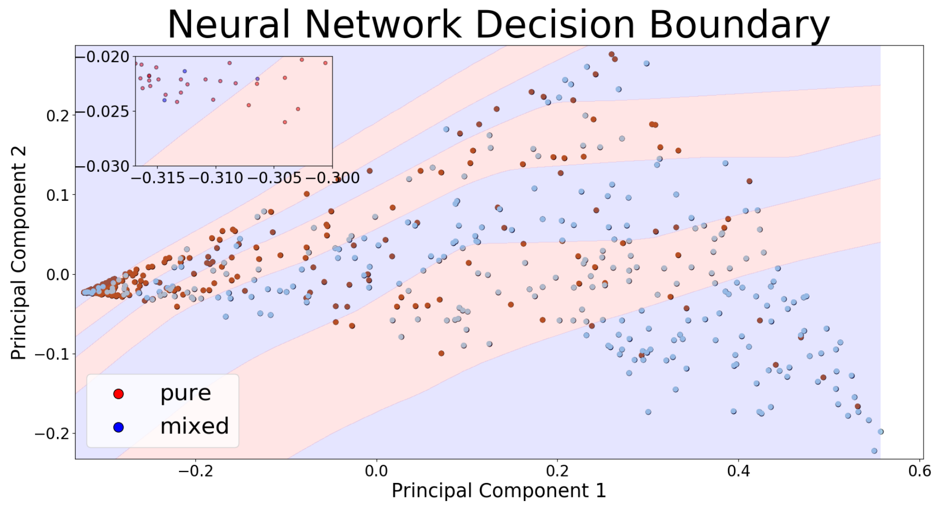

Figure 6: The decision boundary of the neural network shown in figure 1.

The results of the neural network testing suite are shown in figure 7.

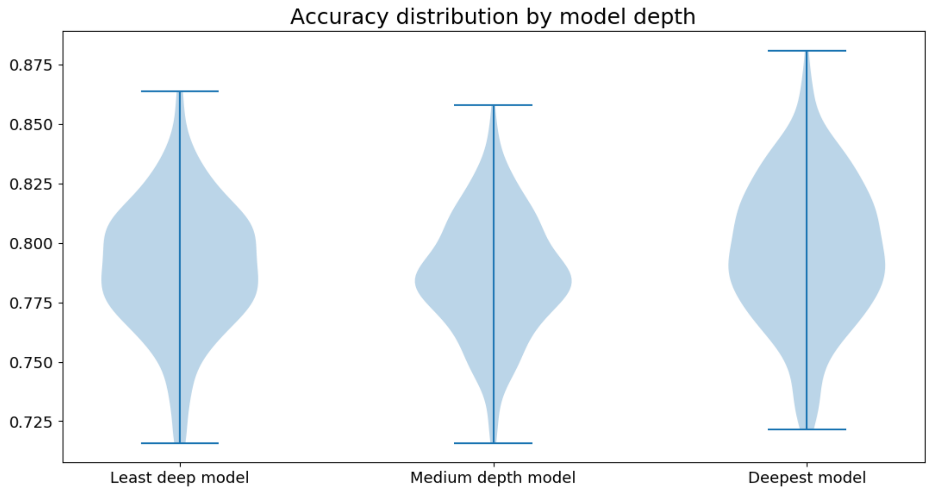

Figure 7: The violin plot shows the distribution of testing accuracy of each neural network type from the testing sweep.

The numeric results of the testing suite are found in the table below.

Model Type	Mean	Standard Deviation
Least Deep (24->12->2->1)	0.7899	0.027
Medium Depth (48->24->4->1)	0.7863	0.025
Deepest (96->48->8->1)	0.7966	0.030

Figure 8 gives the $\text{Accuracy}_{\text{Testing}} - \text{Accuracy}_{\text{Training}}$ of each neural network.

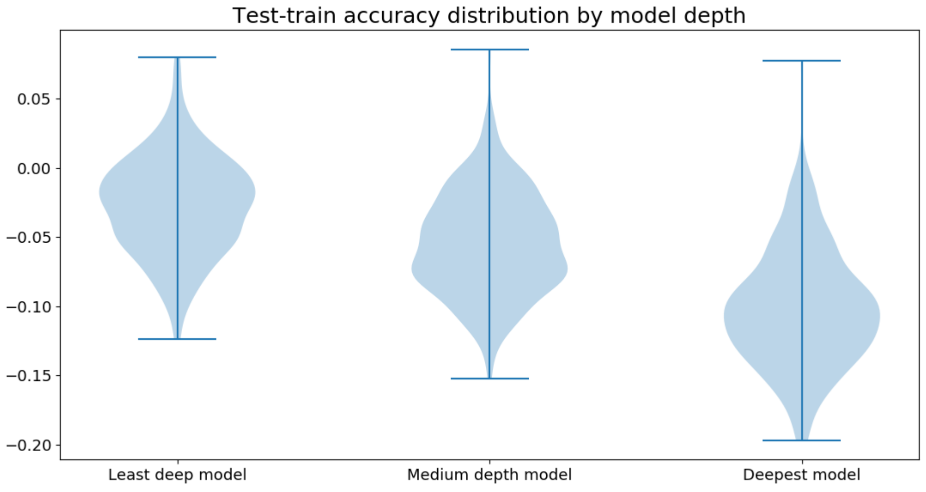

Figure 8: The violin plot shows the distribution of the $\text{Accuracy}_{\text{Testing}} - \text{Accuracy}_{\text{Training}}$ for each neural network.

The numeric results of figure 8 are shown in the table below.

Model Type	Mean	Standard Deviation
Least Deep (24->12->2->1)	-0.027	0.035
Medium Depth (48->24->4->1)	-0.055	0.037
Deepest (96->48->8->1)	-0.102	0.042

3.4 Discussion

From the testing suite in figure 4, the neural network described in figure 2 shows had the highest accuracy on the testing set. This had a mean accuracy of 78% with a standard deviation of 0.028. This is performed better than any of the other models tested in the suite. The neural network did overfit the most, as shown in figure 5. Even with this overfitting the neural network performed the best.

Figure 6 shows the accuracy of the testing suite used on the three neural network structures defined earlier. The figure shows that the deepest model had the best accuracy, but the mean accuracy only shows that here is a 0.67% difference between the least amount of nodes versus the most. In terms of the accuracy there is not much difference between the models. Based on the size of the dataset and the test train split where only 176 samples are in the testing set this difference in accuracy only accounts for a difference of 1 image. This shows that the depth of the neural networks does not change the accuracy much based on dataset provided. Figure 7 shows that the deepest neural network overfits the most, but does not provide that much difference in terms of accuracy from the other two network structures.

4 Related Work

In the initial article utilizing the novel dataset by Khosla et al. [2], the mean accuracy reached 22% when 100 training images were utilized within the SIFT methodology. Liu et al. [5] reached 69% accuracy, with greater success than Khosla et al. [2] due to their partially localized approach. We hope to leverage the bounding boxes and augment the work of Liu et al. [5], possibly improving on their methodology to achieve higher accuracy. Several bounding box approaches are currently being reviewed, although all are endlessly modifiable from simple image segmentation. One is a slight adaptation to a method often employed for training algorithms to images of different shade and dimension called DeepCut. [6] There are several standard variations on the basic DeepCut methodology, which hones in on the nature and dimension of the bounding in a cookie-cutter, traced-out way as opposed to any kind of geometric shape. There was also a Kaggle competition to create the best dog breed classifier of 16 breeds sampled from the Stanford Dog dataset. Here competitors use transfer learning to create the best breed classifier. Competitors were able to get classification scores above 95% in this competition [7]. Microsoft has a dog classifier built on their Bing platform. This platform was used as inspiration for the web app.

5 Code and Dataset

The code can be found in our github repo: `https://github.com/nshenton/CS_251_DogProject`. Please see the Readme in the repo for more information. The Stanford Dogs Dataset can be downloaded from this page: `http://vision.stanford.edu/aditya86/ImageNetDogs/`. Please contact the authors for the mixed dog dataset.

6 Conclusion

Here the methodology described above shows the results of the testing suite, and the neural network is the best method to classify mixed dog breeds based on our method. Of the neural networks tested, there is no real difference between the structures tested besides the overfitting of the deepest model. With the dataset provide it is difficult to distinguish which is the best neural network structure. This allows for some interesting next steps for this model. The first being to increase the dataset small, as the crowdsourced mixed breed data set is still small with 266 images. A larger dataset would be helpful in creating a more robust classifier. Secondly, the hyper parameter of the first convolutional network could be optimized by using transfer learning to better gain better results on the breed classifier. There is a risk of overfitting here, but this could help the accuracy of the second half of the model. This is still a difficult problem as there are a great deal of diversity among all dogs including pure and mixed breed.

This creates a great conversation starter for dog lovers and machine learning enthusiasts alike, where an image can be easily classified as a mixed breed or not, and even some potential breed.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[2] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li, "Novel dataset for fine-grained image categorization: Stanford dogs," in *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, vol. 2, 2011.

[3] V. L. Voith, E. Ingram, K. Mitsouras, and K. Irizarry, "Comparison of adoption agency breed identification and dna breed identification of dogs," *Journal of Applied Animal Welfare Science*, vol. 12, no. 3, pp. 253–262, 2009.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[5] J. Liu, A. Kanazawa, D. Jacobs, and P. Belhumeur, "Dog breed classification using part localization," in *European conference on computer vision*, pp. 172–185, Springer, 2012.

[6] M. Rajchl, M. C. Lee, O. Oktay, K. Kamnitsas, J. Passerat-Palmbach, W. Bai, M. Damodaram, M. A. Rutherford, J. V. Hajnal, B. Kainz, *et al.*, "Deepcut: Object segmentation from bounding box annotations using convolutional neural networks," *IEEE transactions on medical imaging*, vol. 36, no. 2, pp. 674–683, 2016.

[7] Kaggle, "Dog breed identification," 2017.

[8] Google, "Google ai," 2019.

[9] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding box regression with uncertainty for accurate object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2888–2897, 2019.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[12] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.