Prediction of Shelter Pet Photos Popularity based on EfficientNet-B0

EE541 Final Project - Fall 2021

Zijian Cheng, Wenhui Cui, Rongke Zhang

December 13, 2021

1 Introduction

The project is to participate in a competition in an online machine learning community, Kaggle. For this project, we propose to create a model to predict the popularity (pawpularity) of shelter pet photos. We will build a model to rate the cuteness of given photos using the raw images and their metadata. The prediction would be compared with the pet profile's page statistics on PetFinder.my. This program aims to help guide shelter owners to improve the appeal of the pet profiles and to enhance their photo quality.

To address the problem, we will apply a neural network called EfficientNet-B0 to train and evaluate the given pet images. Additionally, mixup data augmentation method and transfer learning method will be used to enhance the prediction accuracy of the model.

2 Method

2.1 Overall Structure

The entire pipeline can be roughly divided into four parts, as shown below (Figure 1). We begin by properly preprocessing the dataset via normalization. Second, we use mixup to improve performance on unseen data through data augmentation. Then, we implement the processed data into a pre-trained EfficientNet model using the transfer learning method. In the end, we use the root-mean-square error (RMSE) as a cost function and train the network to minimize it.

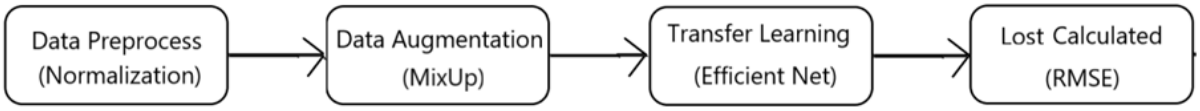


Figure 1. **Overview Flowchart of the Model structure**

2.2 Data Augmentation

We applied several data augmentation strategies to avoid overfitting and to increase its generalization abilities. The methods include randomly resizing the crop, rotating, flipping, and transposing the training data. As a result of some experiments and analysis, we decided to implement the mixup augmentation method to boost our model performance further.

The mixup method trains a neural network on convex combinations of examples and their labels. Mixup can therefore perform regularization on the neural network to favor simple linear behavior between examples. Moreover, it increases the robustness of generative adversarial networks to adversarial examples, reduces the memorization of corrupt labels, and stabilizes the training of generative adversarial networks[1]. The formula of mixup is given below:

$$\hat{x} = \lambda x_i + (1 - \lambda) x_j, \qquad where\ x_i, x_j\ are\ raw\ input\ vectors$$

$$\hat{y} = \lambda y_i + (1 - \lambda) y_j, \qquad where\ y_i, y_j\ are\ one - hot\ label\ encodings$$

Randomly select two pictures, and then generate a parameter of [0,1] to add the two pictures together, so the new sample becomes a "watermark" one on top of the other. Mixup extends the training distribution because linear interpolations of feature vectors should lead to linear interpolations of associated targets.

For the project, we choose alpha=0.2 and let lambda be a random variable in Beta(0.2, 0.2) distribution.

2.3 EfficientNet Baseline Architecture

The number of parameters grows quickly as neural networks go deeper, which would make the computation heavy and difficult. To reduce the number of parameters, we adopt CNN to do our image classification, specifically the EfficientNet model baseline.

EfficientNet architecture is relatively small among neural networks but can achieve very high accuracy on ImageNet. By balancing network depth, width and resolution, EfficientMet achieves better performance with a relatively smaller number of parameters.

Unlike conventional networks that arbitrarily scales the depth, width and resolution, EfficientNet uniformly scales all three with a fixed compound scaling coefficient. Figure 2 illustrates the EfficientNet structure given by the creator of this model.

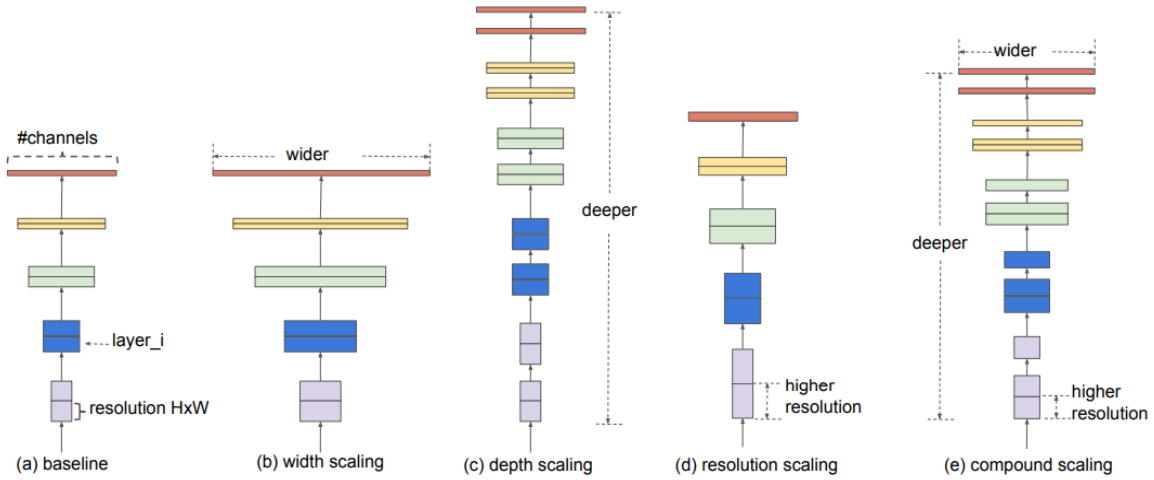


Figure 2. **Model Scaling** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio[2].

In our project, we used EfficientNet-B0 as our baseline model. The structure of EfficientNet-B0 is listed as in Table 1[2]:

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Table 1. **EfficientNet-B0 Structure** Each row describes a stage i with $\widehat{Li}$ layers, with input resolution $\widehat{Hi} \times \widehat{Wi}$ and output channels $\widehat{Li}$[2]

Among all EfficientNet models, EfficientNet-B0 has the smallest number of layers and parameters, which makes it easily scalable to produce efficient and accurate results. According to the creator\cite{Tan2019}, this model can achieve an accuracy above 82\% with compound scaling. The accuracy of each scaling method on ImageNet is shown in Figure 3 and Table 3 below.

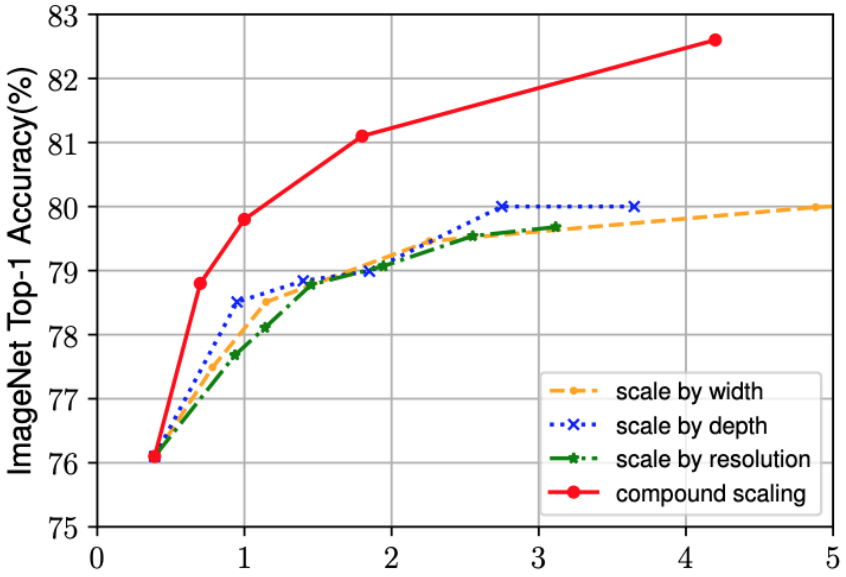


Figure 3. **Scaling Up EfficientNet-B0 with Different Methods**[2]

Model	FLOPS	Top-1 Acc.
Baseline model (EfficientNet-B0)	0.4B	76.3%
Scale model by depth (d=4)	1.8B	79.0%
Scale model by width (w=2)	1.8B	78.9%
Scale model by resolution (r=2)	1.9B	79.1%
Compound Scale (d=1.4, w=1.2, r=1.3)	**1.8B**	**81.1%**

Table 2. **Scaling Up EfficientNet-B0 with Different Methods**[2]

2.4 Transfer Learning

Transfer learning is a method of applying knowledge gained while solving one problem to another related problem. When learning to recognize cats, for example, knowledge gained can be applied to understanding dogs. Since the EfficientNet network on this problem requires a massive amount of computation and time to develop, Transfer learning can save the cost and provide a considerable jump inability on related problems.

We apply transfer learning to boost our model performance. Specifically, we adopt an efficient net-b0-noisy-student model pre-trained on ImageNet data and fine-tune this model on our task. We first load the pre-trained weights and then randomly initialize the final prediction layer. Then we start training on our animal image data.

3 Implementation

3.1 Data Processing

3.1.1 Dataset

The purpose of this project is to determine engagement with a pet's profile by analyzing the pet's photo. The dataset is 1.04 GB GB and the training data consists of a .csv file and image files. The .csv file consists of training set photos' metadata and their pawpularity score.

The image file (Figure 4) consists of around 10000 training images of the form {id}.jpg, where {id} is a unique Pet Profile ID. Pet images are colored pictures of assorted dimensions and sizes. The testing data has a similar format of the training data, with 6800 pet photos similar to the training set photos. The focus of our research is mainly on applying images to an EfficientNet network.

Figure 4. **Example Training Data** Each picture comes with a unique pet profile id and popularity score

Metadata (Figure 5) include each photo ID in the training set and the target, as well as the photo's pawpularity score. The metadata contains labels of 1(Yes) or 0(No) for 12 features of each training set photo. And the Id column gives the photo's unique Pet Profile ID corresponding to the photo's file name.

	Id	Subject Focus	Eyes	Face	Near	Action	Accessory	Group	Collage	Human	Occlusion	Info	Blur	Pawpularity
0	0007de18844b0dbbb5e1f607da0606e0	0	1	1	1	0	0	1	0	0	0	0	0	63
1	0009c66b9439883ba2750fb825e1d7db	0	1	1	0	0	0	0	0	0	0	0	0	42
2	0013fd999caf9a3efe1352ca1b0d937e	0	1	1	1	0	0	0	0	1	1	0	0	28
3	0018df346ac9c1d8413cfcc888ca8246	0	1	1	1	0	0	0	0	0	0	0	0	15
4	001dc955e10590d3ca4673f034feeef2	0	0	0	1	0	0	1	0	0	0	0	0	72

Figure 5. **Metadata Screenshot**

In addition, from the distribution of pawpularity scores (Figure 6), we observe that a large number of entries have pawpularity scores of 100, which skews the distribution of pawpularity scores. Additionally, a small bump of pawpularity scores close to zero is also evident. Apart from this, the data seems to roughly follow a gamma distribution.

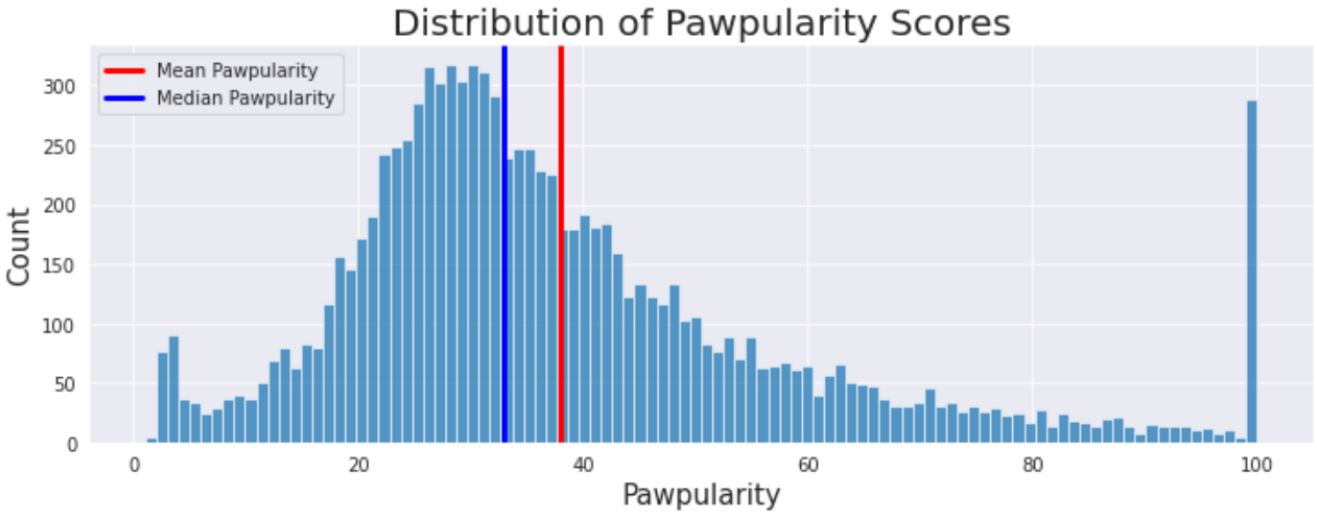


Figure 6. **Popularity Score Distribution**

3.1.2 Normalization

Normalization was applied to the dataset. Normalization is the process of changing the range of pixel intensities in image processing. It is sometimes referred to as contrast stretching or histogram stretching. Normalization transforms an n-dimensional image I: $\{X \subseteq Rn\} \; - \rightarrow \{Min,..,Max\}$ with intensity values in the range (Min, Max)into a new image In: $\{X \subseteq Rn\} \; - \rightarrow \{newMin,..,newMax\}$ The linear normalization of a grayscale digital image is performed according to the formula [3]:

$$In = (I - Min)\frac{newMax - newMin}{Max - Min} + newMin \quad (1)$$

To enhance the generalization ability of our model and increase performance on the unseen testing data, various strategies of data augmentation were applied, including random resize crop, rotating 90 degrees, flipping, transposing, as well as normalizing to the training data. To ensure that each input pixel has a similar data distribution, the original image dataset from 720*405*3 got resized to 512*512*3. An example of the normalization process is presented below (Figure 7).

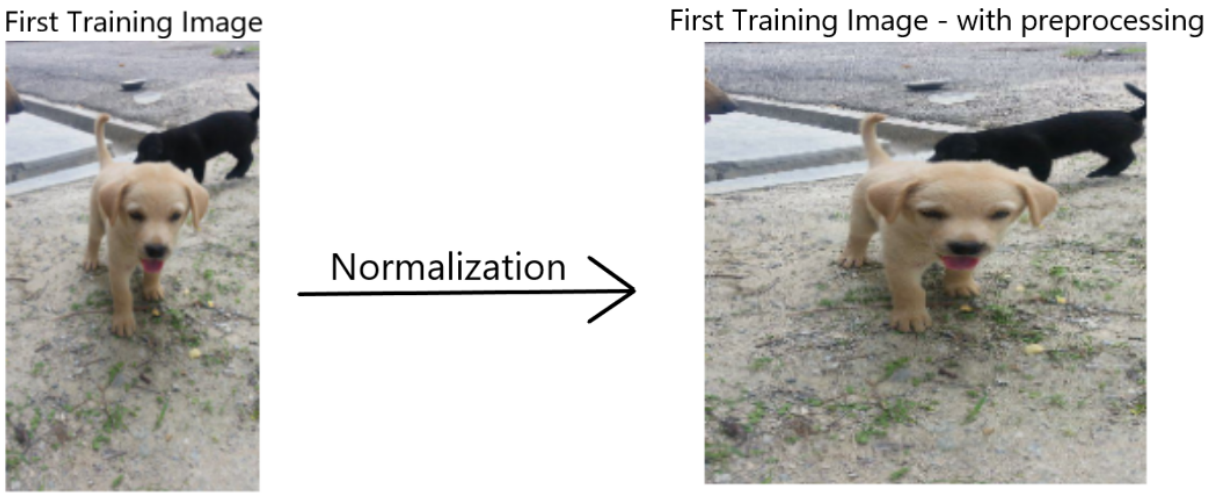


Figure 7. **Normalization**

3.2 Loss Function

As mentioned above, the Root Mean Square Error (RMSE) was calculated as the model cost function. RMSE is a standard way to measure the error of a model in predicting quantitative data. Formally it is defined as follows:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y\hat{}i - yi)^2}{n}} \quad (2)$$

Where $y\hat{}1$, $y\hat{}2$,..., $y\hat{}n$are predicted values and $y1$, $y2$,..., yn are observed values, n is the number of observations.

3.3 Training Details

The model utilized Adam optimizer instead of stochastic gradient descent algorithm. Instead of a fixed gradient, Adam optimizer uses an adaptive gradient algorithm so that the learning rate adapts to potentially sparse gradients[4]. This is especially useful because the pet images provided may be drastically different. To adjust the learning rate, we applied a cosine annealing learning schedule during training. Since we adopted a pre-trained model(EfficientNet-B0), the gradients could be small with pre-trained weights, therefore we started our training rate at a relatively low level(1e-4).

Due to limited computation and memory resources, we made the batch size 16. We trained the model for 10 epochs, and it costs approximately 90 minutes on an Nvidia TITAN Xp GPU.

4 Experiment and Results

4.1 Cross-Validation

In order to avoid overestimating the model performance, we used 5-fold cross-validation to evaluate our model. We shuffled our dataset and randomly picked pictures to test the accuracy of our model, and took the average over fivefold to get the final RMSE.

4.2 Fivefold result based on different resolutions

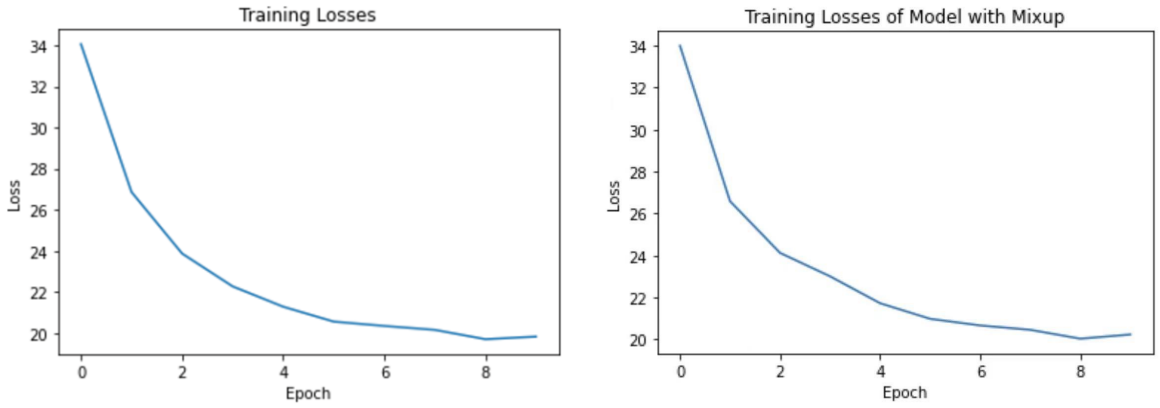
The data images are given in assorted sizes. When deciding on the image resolution, we attempted two different images sizes, 512*512 and 256*256. We decided to use the higher resolution based on the model performance (Table 3).

Image resolution	Average RMSE over five folds
256x256	**23.2684**
512x512	**20.7509**

Table 3. **5-Fold Cross-Validation Baseline Model RMSE Average**

4.3 Fivefold result based on 512x512 resolution with and without mix-up augmentation

Since the testing images for the competition are not available to us, we must ensure that our model works on generalized data. We compared the results before and after mixup augmentation as below(Figure 8 and Table 4). After adding in augmentation, our model got a smaller RMSE.



(a) (b)

Figure 8. **Training loss over epochs** (a) Training losses of the model without mixup augmentation (b) Training losses of the model with mixup augmentation.

Augmentation	Average RMSE over five folds
No	**20.7509**
Yes	**20.3429**

Table 4. **Average RMSE Before and After Augmentation**

4.4 Final leaderboard result

The submission score of RMSE is 19.82024. The highest leaderboard ranking is 1896/2851 at the time of submission (Figure 9).

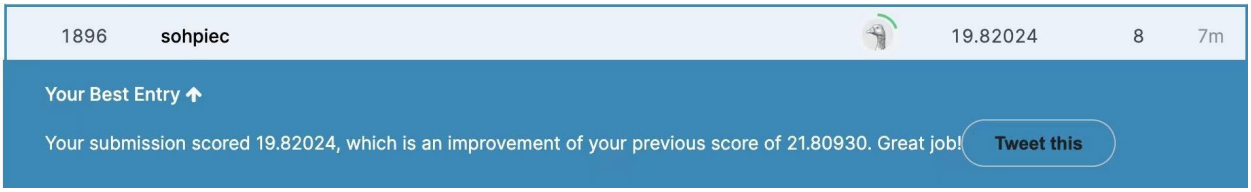


Figure 9. **Leaderboard Result**

5 Conclusion

In this project, we created a convolutional neural network based on EfficientNet-B0 and loaded the weights. Then, we trained the model based on the training data. Afterward, we evaluated the performance of the model on the testing data with 5-fold cross validation. With the well-trained model, we performed inference for the unseen test data and generated a submission.csv file using a jupyter notebook. Finally, the competition website evaluated the performance of our model and returned a RMSE score of 19.82024. The highest leaderboard ranking is 1896/2851 at the time of submission.

6 Challenges and Future Work

6.1 Challenges

One challenge that we met is that When performing cross-validation, after running the first fold, we met cuda out of memory problem. It cannot allocate more memory for the second fold. We solved this problem by deleting the model after each fold, and using torch.no_grad() during validation to save memory. Additionally, we cannot try other versions of EfficientNet due to time and computation resources limits.

6.2 Future Work

As previously mentioned, the competition offers both image data and metadata, but we only use the image data right now. Therefore, we will combine both data sets in our future work in order to get a better result by utilizing as many data points as possible. As shown below (Figure 10), we first extracted the image embeddings from EfficientNet. Afterward, we apply them with metadata into a regression model. This combined model can then provide us with a prediction.

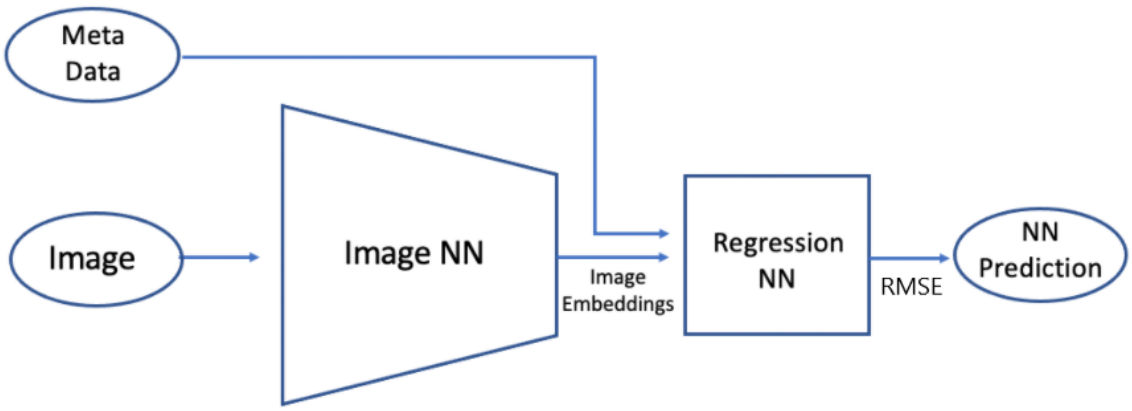


Figure 10. **Proposed Future Work Structure Flowchart**

References:

[1] Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond Empirical Risk Minimization. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. https://arxiv.org/abs/1710.09412v2

[2] Tan, M., & Le, Q. v. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *36th International Conference on Machine Learning, ICML 2019, 2019-June*, 10691–10700.

[3] "Normalization (Image Processing)." *Wikipedia*, Wikimedia Foundation, 20 May 2021, https://en.wikipedia.org/wiki/Normalization_(image_processing).

[4] Kingma, D. P., & Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. https://arxiv.org/abs/1412.6980v9