Reference code link : **Link: https://www.kaggle.com/reppy4620/fast-ai-efficientnetb0-5fold-training**

We adopted EfficientNet (EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks ) as our backbone architecture. (cite this paper, add github repo link. https://github.com/lukemelas/EfficientNet-PyTorch)

Introduce Efficientnet: (from the paper or from the internet.) Its architecture, its improvement compared to other models.

(Also introduce efficientnet-b0-noisy-student, because this is the exact version we used. https://towardsdatascience.com/efficient-nets-with-noisy-student-training-5ac6e239ff14)

We chose EfficientNet because of its superior performance demonstrated in various computer vision tasks.

Our model predicts the popularity score and minimizes the root mean squared error (RMSE) between the predictions and the ground-truth scores. So, it can be viewed as a regression problem.

# Figure: EfficientNet Model Architecture

# Transfer Learning:

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

# Data Augmentation:

To enhance the generalization ability of our model and increase performance on the unseen testing data, we applied various strategies of data augmentation. We applied random resize crop, random rotate 90 degrees, flip, transpose, as well as normalization to the training data.

We only apply normalization to the testing data.

Furthermore, we implemented mixup augmentation method to further boost our model performance.

*Introduce mixup (cite the paper:* mixup: BEYOND EMPIRICAL RISK MINIMIZATION

*):*

x=lambda\*x1+(1-lambda)x2

y=lambda\*x1+(1-lambda)y2

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.

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

# Detailed training settings:

We perform 5-fold cross-validation to test our model performance on the available data.

We applied Adam optimizer. (maybe introduce adam optimizer)

Applied cosine annealing learning schedule to adjust learning rate during training.

Our initial learning rate is set to be 1e-4, since we adopted a pretrained efficientnet-b0 model. The gradients could be small with the pretrained weights, so we need to start with a small learning rate compared to train from scratch.

We use an image size of 512 x 512, because higher resolution will result in better predictions.

The batch size is 16 due to memory limits.

We trained the model for 10 epochs, and it costs approximately 90 minutes on a Nvidia TITAN Xp GPU.

The train.csv

Background pattern

Description automatically generated with low confidence

Training Session Screenshot:

# A screenshot of a computer Description automatically generated with medium confidence

Chart, line chart

Description automatically generated

# Experiments

1. **5-fold cross-validation baseline model RMSE:**

**(Image size=512x512)**

**Average RMSE over 5 folds: 20.7509;** 5folds [21.4824, 20.7832, 20.6243, ,20.6243, 19.6707]

1. **5-fold cross-validation baseline model RMSE:**

**(Image size=256x256)**

**Average RMSE over 5 folds: 23.2684;**

From here, we know that the larger the image size is, the better the results are.

1. **With Mixup augmentation: 5-fold cross-validation average RMSE:**

**Make a table showing the comparison between the results**

# Inference for Submission

We construct a model and load the weights trained on all the training data. We perform inference on the well-trained model and get the final predictions for the unseen testing data.

Then we generate a submission.csv file and put the predictions into the file. We have no access to the testing data of the competition.

According to the competition requirements, we need to submit a jupyter notebook that generates the submission file. And the notebook must be runnable without internet.

So I uploaded our well-trained model and upload the downloaded packages required by our code to Kaggle dataset. So the notebook can run without internet.

**Our RMSE Loss on testing data is:** 21.80930

**Leaderboard rank: 2706/2851**

**Discussion:**

1. **I met problems when submitting my notebook to Kaggle. It does not allow internet, but I need to pip install some packages, which requires internet connection. So I managed to first download the packages to local machine, and then upload to Kaggle dataset, and then import from dataset.**
2. **Due to computation resources limits, I cannot try other versions of efficientnet. And a larger batch size might improve the model performance, too.**
3. **When performing cross-validation, after running the first fold, I met cuda out of memory problem. It cannot allocate more memory for the second fold. I solved this problem by deleting the model after each fold, and use with torch.no\_grad() during validation to save memory.**