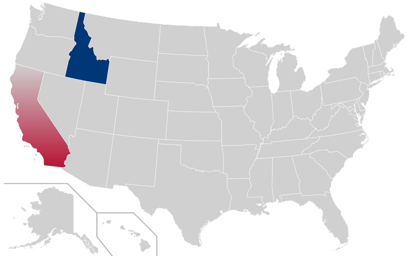
iWildCam
Kaggle Final Project

Assigned By: Eric Kulikov, Eyal Zvi

About The Data

The data is a combination of images from the iNaturalist datasets 2017- 2019.
Divided into:

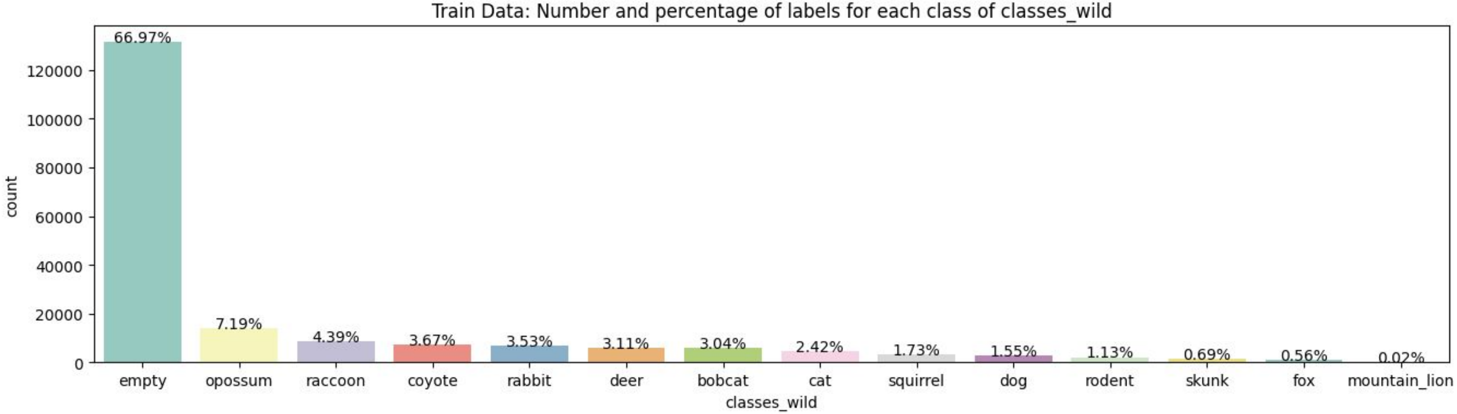
- Train Set: 196,157 images from 138 locations in Southern California.
- Test Set: 153,730 images from 100 locations in Idaho.



There are 23 labels for prediction where ‘0’ represents an absence of an animal in a picture and ‘1’ - ‘22’ represent different types of animals (deer, wolf, squirrel, etc…)

Data Selection

While observing the data we have noticed that there is a big percentage of ‘empty’ labeled images. Moreover, only 14 of the desirable 23 classes are even represented in the normal training data.



These are quite the issues, having an over-representation of a certain class may result in skewed results, and the same goes for an under-representation, let alone no representation at all.

Code(Plot): Gabriel Preda – iWildCam 2019 EDA and Prediction – kaggle.com/code/gpreda/iwildcam-2019-eda-and-prediction

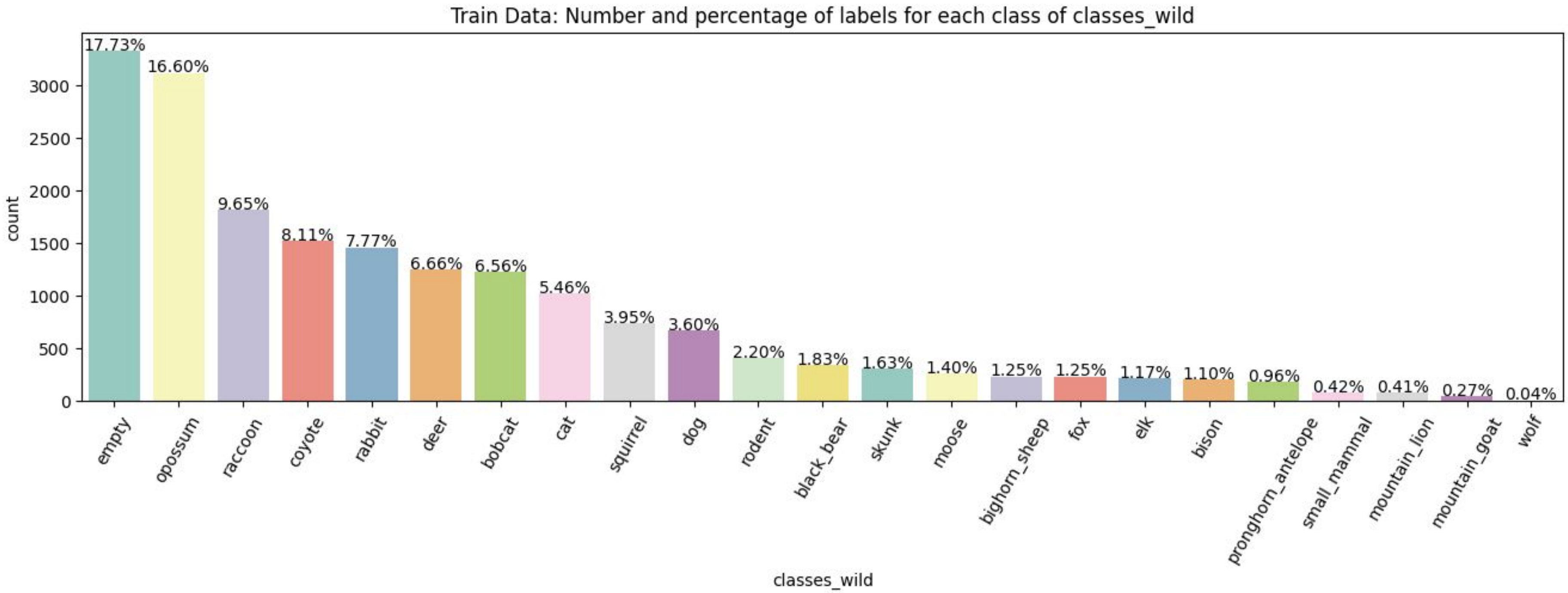
Data Selection

In order to solve these problems we have come up with two elegant solutions:

1. Outsourcing – As stated in the rules, new images from similar challenges could be added. These images provide us with a representation for the missing classes and help stabilize the skewed data.

2. Undersampling – In order to minimize the effect of representation, we aspire to train our network on a data that is close to being uniformly distributed.
   It is worth mentioning that estimating the real distribution of these animals can be very difficult, after trial and error we have found that randomly replacing 'empty' labeled images with new classes can be of good practice.

Data Selection

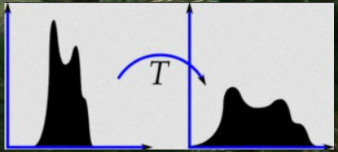


Data Preprocessing – Selection

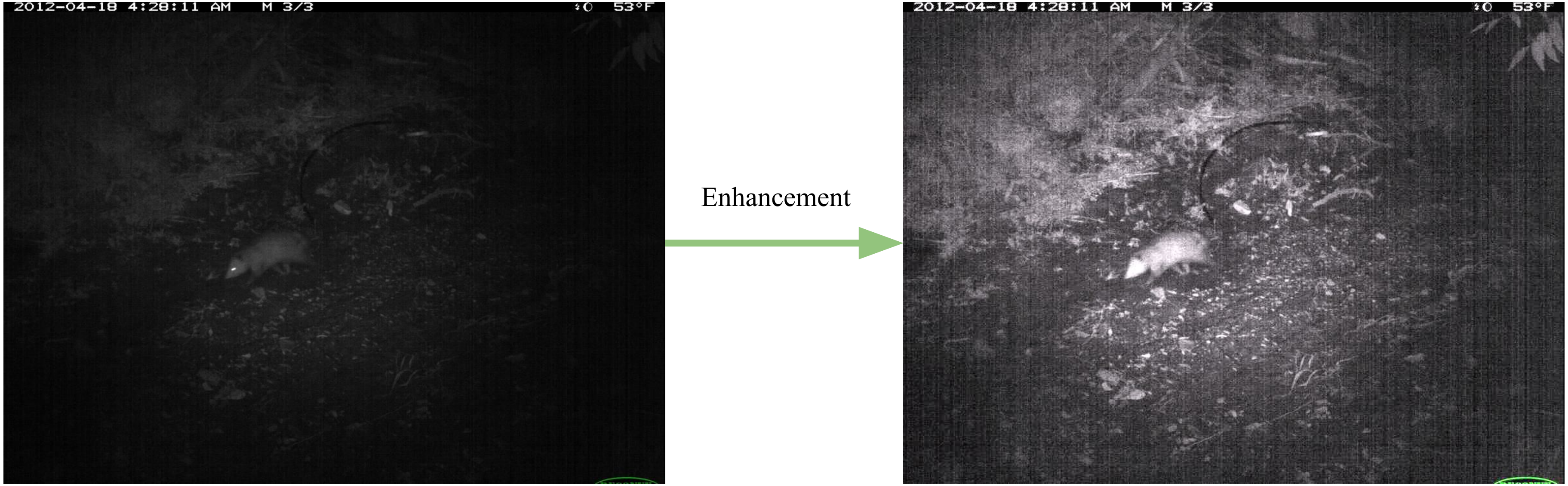
iNaturalist dataset is partially uncurated, we have manually filtered out images that may classify a certain animal to some aspect, but will not be helpful in the training process.

"Wolf"

Data Preprocessing – Enhancement



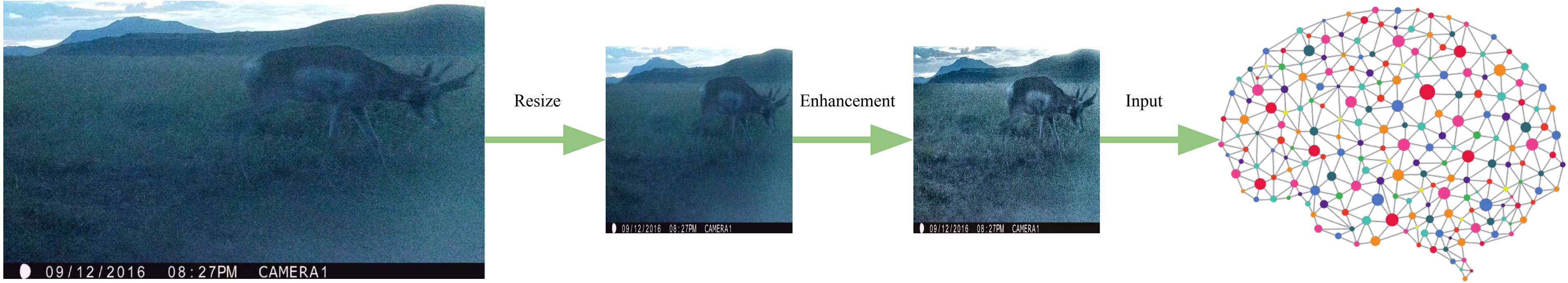
We have applied a <u>CLAHE</u> (**C**ontrast **L**imited **A**daptive **H**istogram **E**qualization) filter and a <u>White Balance</u> filter to all the images in order to improve the visibility of different animal features.



Code: Chanran Kim – Image Pre-processing for Wild Images – kaggle.com/code/seriousran/image-pre-processing-for-wild-images

Data Transformation

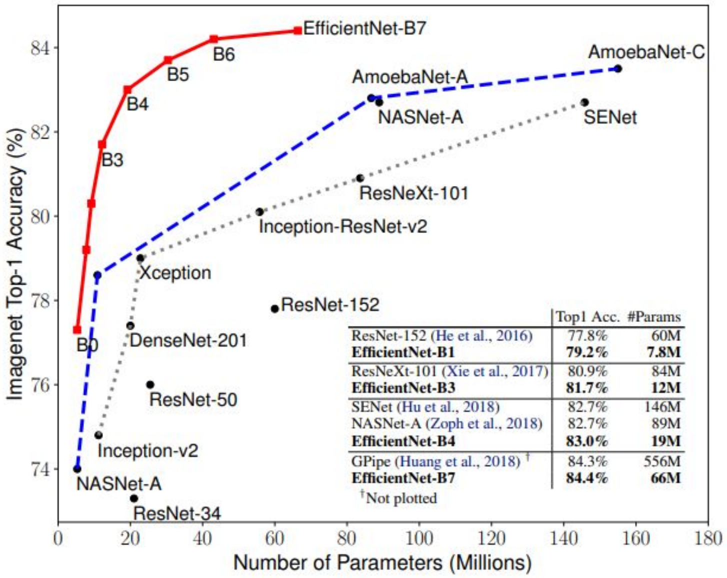
- Grayscale images were transformed from a one dimensional coloring vector to a three dimensional coloring RGB vector.
- All images were then resized to fit a fixed 224x244x3 size that will be the input size of our neural network for this classification task.



Neural Network – EfficientNetB0

EfficientNetB0 is a pre-trained neural network that was trained on over one million images from the ImageNet database. It is able to categorize images into 1,000 different object classes, **including animals** as well as common objects like keyboards, mice, and pencils.

As a result, the network has accumulated knowledge of extensive feature representations that can be applied to a wide range of images, including our training.



Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *In International conference on machine learning*, pp. 6105-6114. PMLR, 2019.

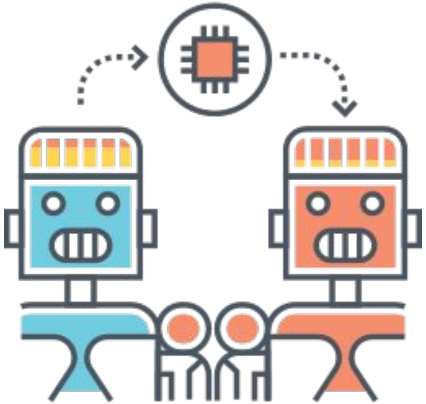
Neural Network – EfficientNetB0 Cont.

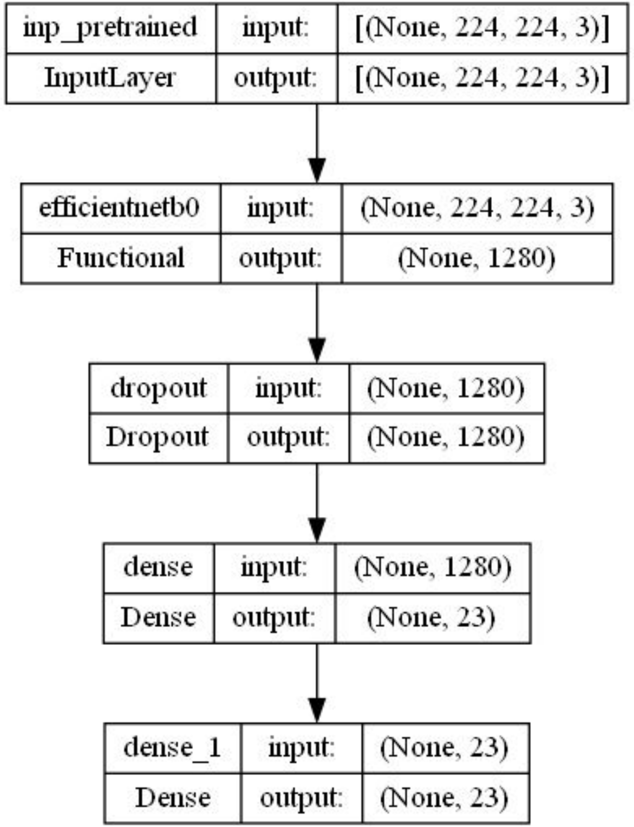
Why EfficientNetB0? → Smaller model with fewer parameters and therefore faster computation.

How to adapt EfficientNetB0 to our problem? → Transfer Learning (Can help with class over-representation)

How to overcome overfitting? →

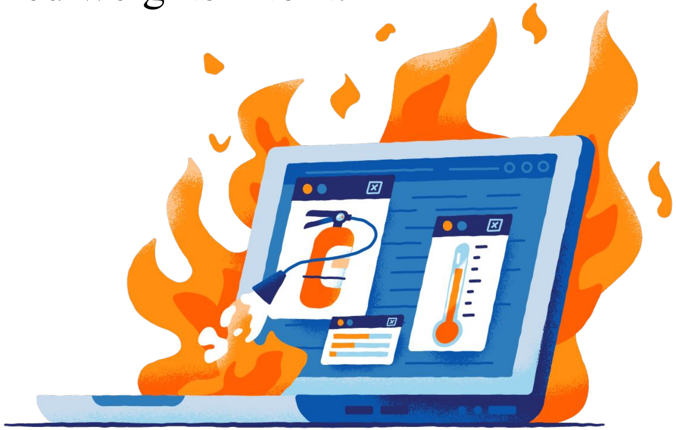
- Dropout Layers
- Simple Model
- Early Stopping – Patience
- 5-Fold CV





Training

1. Training the top layer
    1.1. Get the base model (EfficientNetB0) and load pre-trained weights into it.
    1.2. Freeze all layers of the base model.
    1.3. Stack a new model on top of the new base model.
    1.4. Train the base model only.



2. Training the base model – Fine-tuning
    2.1. Wait until top layers have converged
    2.2. Unfreeze the base model
    2.3. Retrain the whole model end-to-end with a low learning rate.

Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson. "How transferable are features in deep neural networks?." Advances in neural information processing systems 27 (2014).

Transfer Learning in Keras: keras.io/guides/transfer_learning/

Workflow

Preprocessing

- Selection: Filter Images
- Transformation: Resize to 224
- Enhancement: CLAHE + WB

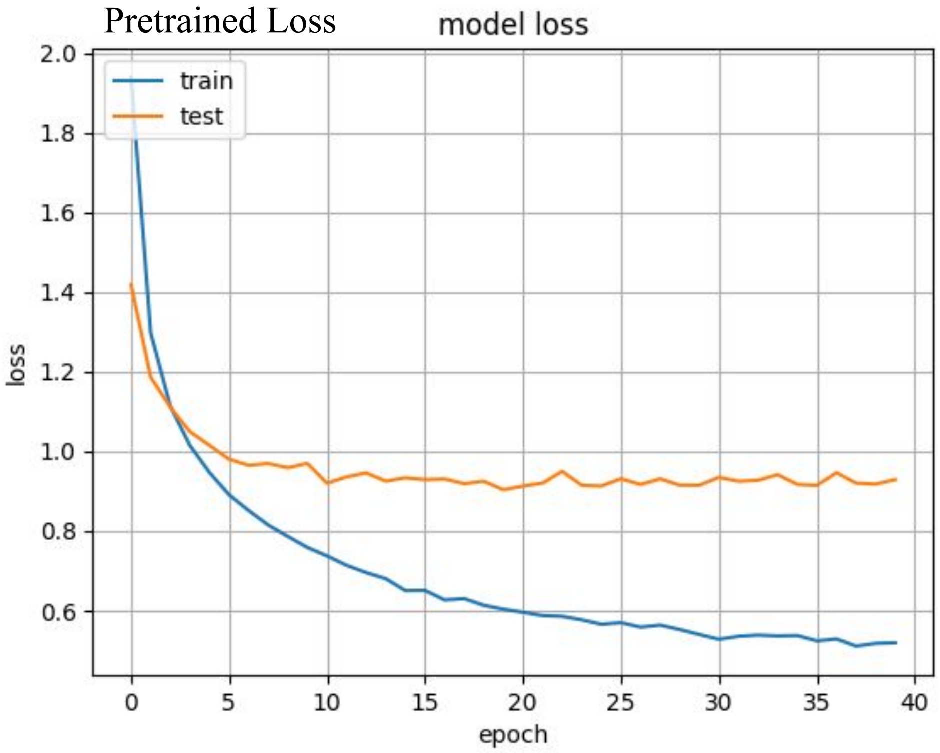
Top Layer Training

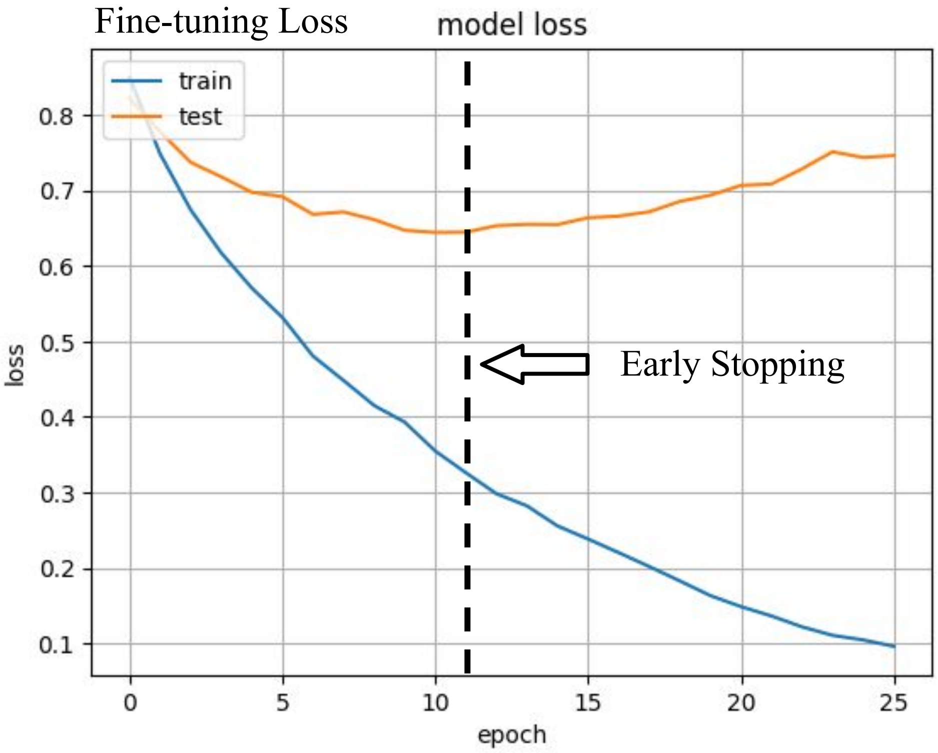
- Pre Trained EfficientNetB0
- Attach a FC layer with 23 nodes
- Train only the top layers

Fine-tuning

- Unfreeze the pre-trained model
- Train end-to-end with low learning rate

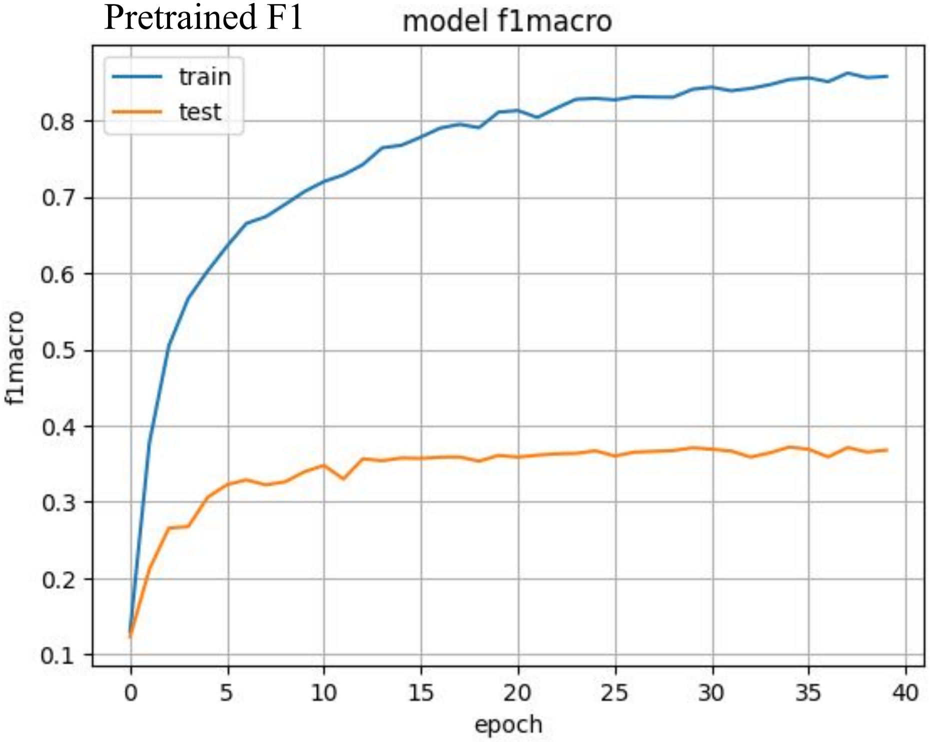
Evaluation – Loss



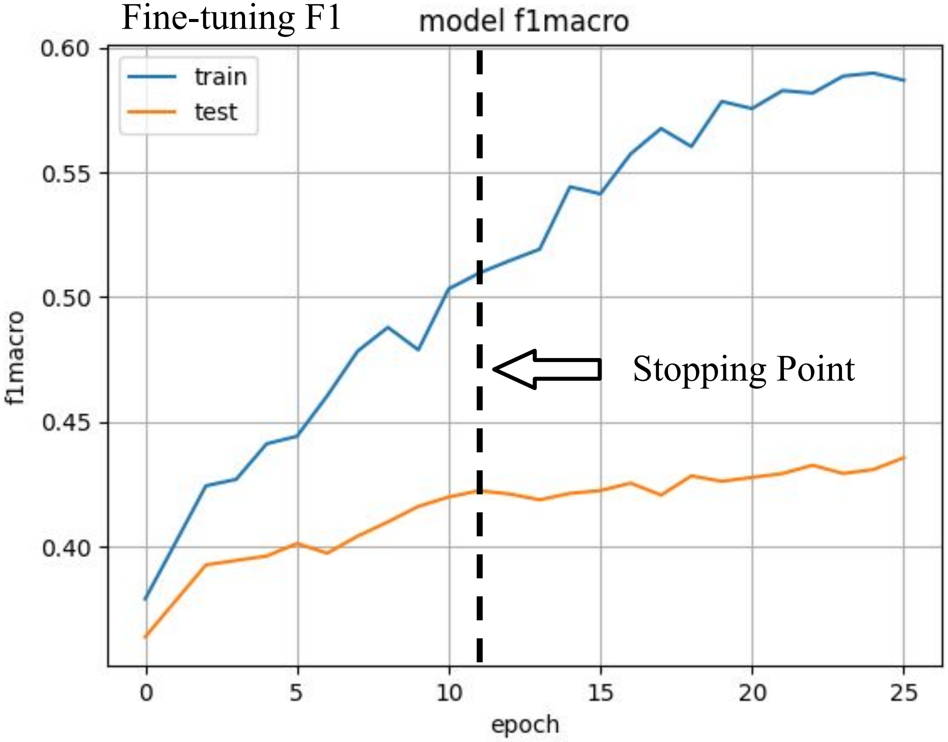


Evaluation – Macro F1 Score (Train data)

Pretrained F1



Fine-tuning F1



Evaluation – Kaggle

YOUR RECENT SUBMISSION

submissionFinalFIXED.csv

Submitted by Road Kill · Submitted 19 hours ago

Score: 0.125

Private score: 0.117

↓ **Jump to your leaderboard position**

Thank You For Listening!

Questions?