Project Report: Flower Classification with TPUs

CS 795/895 Practical Machine Learning (Spring, 2020)

Kritika Garg
Department of Computer Science
Old Dominion University
Kgarg.kritika@gmail.com

***Abstract*— Flower Classification with TPUs is a Kaggle ongoing competition which presents the challenge of classifying 104 types of flowers using TPU. In this report, we describe our approach to flower classification using Deep Learning models.**

I. Introduction

There are over 400,000 different types of flowers in this world. The Kaggle's Flower Classification with TPUs competition presents a challenge to its participants to build a machine learning model that identifies the type of flowers in a dataset of images and classifies the 104 types of flowers using TPU. Tensor Processing Unit (TPU) are powerful AI accelerator application-specific integrated circuits (ASIC) specialized in deep learning tasks. The kaggle provides three dataset (train, validation, test) across four different resolution (512, 331, 224, 192).The dataset provided by kaggle is in the .tfrec format, is a container format frequently used to group and shard data data files for optimal training performance. The main challenge of this competition is to use TPU to perform these classification.

II. Methodology

The details of our approach is presented in this section. In this work, we will see the use of different deep learning models to perform the image classification.

A. DATA VISUALIZATION

I used the visualization functions in the getting started kernel provided by kaggle. Fig 1. shows the visualization of the small sample from the training dataset.

Fig. 1. labelled images in training dataset.

The training data contains 12753 images, while validation data had 3712 images. I explored the distribution of these images among the 104 classes. Fig2. a) shows the distribution of training and validation dataset.

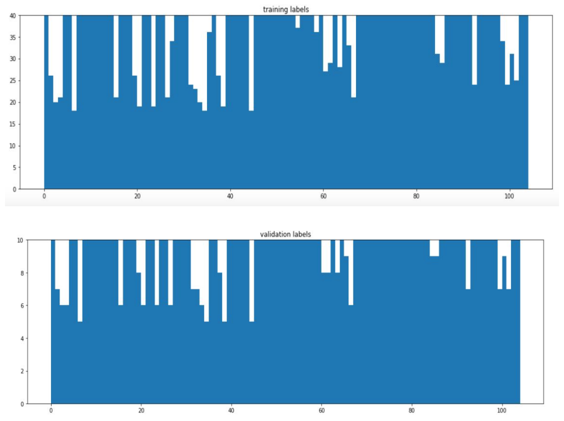


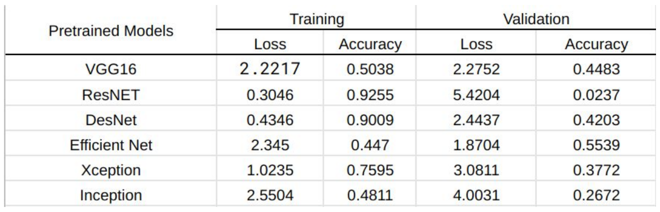
Fig. 2. Distribution of training and validation dataset

It is evident from these images that the data is unbalanced. I observed inconsistencies in the classes. Some of the flowers like "pink primrose" had 272 images while flowers like "sweet pea" only had 21 images in the training dataset.

B. BASELINE MODEL

Keras ships out-of-the-box with five Convolutional Neural Networks that have been pre-trained on the ImageNet dataset: VGG, ResNet, Inception, and Xception. I used these five neural networks along with EfficientNet to construct the baseline models. I used the 192x192 resolution training dataset to train these models and evaluated them on the validation dataset. To construct these models, I used keras sequential modeling with GlobalAveragePooling2D layer and Densely connected neural network layer with softmax activation function. I used Adam algorithm to optimize these models and sparse_categorical_crossentropy to compute the loss. To compute the accuracy, I used sparse_categorical_accuracy.

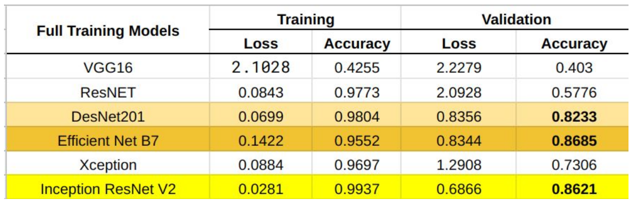
To construct these baseline models, My first approach was to construct transfer learning baseline models. The transfer learning models reuse pre-trained imagenet keras models for its base layers. Fig 3. shows the loss and accuracy of the transfer learning models across training and validation.



Pretrained Models	Training		Validation	
	Loss	Accuracy	Loss	Accuracy
VGG16	2.2217	0.5038	2.2752	0.4483
ResNET	0.3046	0.9255	5.4204	0.0237
DesNet	0.4346	0.9009	2.4437	0.4203
Efficient Net	2.345	0.447	1.8704	0.5539
Xception	1.0235	0.7595	3.0811	0.3772
Inception	2.5504	0.4811	4.0031	0.2672

Fig. 3. Loss and accuracy of transfer-learning baseline models

The transfer learning models performed very poorly i.e. the validation accuracy was low for all the models. My next approach was to fully train these models.



Full Training Models	**Training**		**Validation**	
	Loss	**Accuracy**	**Loss**	**Accuracy**
VGG16	2.1028	0.4255	2.2279	0.403
ResNET	0.0843	0.9773	2.0928	0.5776
DesNet201	0.0699	0.9804	0.8356	**0.8233**
Efficient Net B7	0.1422	0.9552	0.8344	**0.8685**
Xception	0.0884	0.9697	1.2908	0.7306
Inception ResNet V2	0.0281	0.9937	0.6866	**0.8621**

Fig. 4. Loss and accuracy of fully trained baseline models

Some of the fully trained baseline models performed very well with the highest accuracy of 0.86. As seen in Fig 4, The best performing models were Inception ResNet V2, DenseNET 201, and Efficient NET B7 with accuracy of 0.8621, 0.8233, and 0.8685. This encouraged me to explore these models further and to improve their accuracies.

C. DATA AUGMENTATION

Data augmentation is a common technique to improve results and avoid overfitting. It also enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. This technique seemed to be popular among kagglers participating in this competition, so I tried to implement this technique in my work. I implemented various augmentation techniques such as spatial level transform, pixel level transform, and other miscellaneous transformations such as transform rotation, transform shift, transform shear, and transform zoom. The combination of these augmentation techniques can be seen in Fig 5. where you can see the various transformations of "morning glory" flowers.

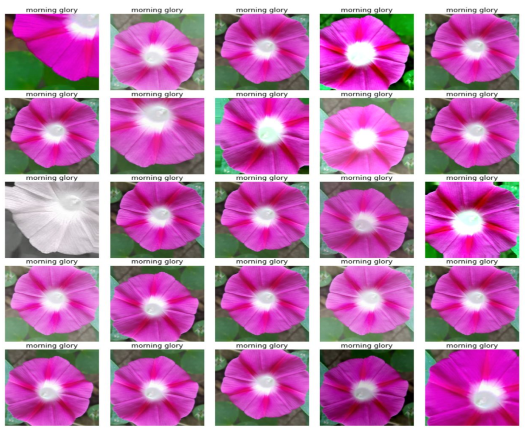


Fig. 5. Visualization of data augmentation transformation on "morning glory" flower image.

I explored the effect of these transformations on validation accuracy. On applying all of the above mentioned techniques together, the validation accuracy decreased by 80%. I examined the different combinations of transformations to select the one that had the positive effect on the validation. The transformation techniques used are spatial level transformations such as random flip left right, random flip updown and random crop.

D. LEARNING RATE SCHEDULER

Learning Rate is the amount of change to the model during each step of this search process, or the step size and provides perhaps the most important hyperparameter to tune for the neural network to achieve good performance. When training deep neural networks, it is often useful to reduce learning rate as the training progresses. To implement this I used a custom learning rate scheduler. Learning rate schedules seek to adjust the learning rate during training by reducing the learning rate according to a predefined schedule. To configure the LR scheduler, I used 0.0001 as starting learning rate, 0.00001 as min learning rate, and 0.8 as LR exponential decay.

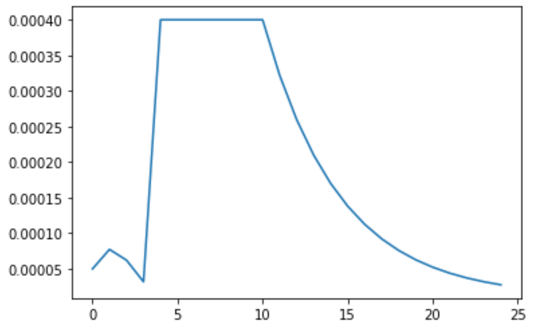


Fig. 6. The learning rate varying with respect to epochs

The Fig6. shows the variation in learning rate with increase in epoch. The LR varies from 4.95e-05 to 0.0004 to 2.72e-05.

E. MODEL

In this work, I implemented different Convolutional neural networks (CNNs) to perform the image classification. After evaluating the six different baseline models, I selected the models with the best validation accuracies. The next sections of this report introduce these models and show the detailed approach of improving these model's performance. The models used are Inception ResNet V2, DenseNET 201, Efficient NET B7. To implement these models, I used Keras Applications which is deep learning models that are made available alongside pre-trained weights.

1. DenseNet-201

DenseNet-201 is a convolutional neural network that is 201 layers deep. To implement this model, I used Keras Applications that contains the DenseNet models, with weights pre-trained on ImageNet. The model was constructed using keras sequential model with GlobalAveragePooling2D layer and Densely connected neural network layer of 104 length with softmax activation function. The loss was computed using sparse categorical cross-entropy and the accuracy was computed using sparse categorical accuracy. Adam algorithm with constant learning rate of 0.0001 was used for optimization.

To train this model, I used 20 epochs with 128 steps per epoch. For Initial training, I used 224 input image size as higher size was causing the over exhaustion of the resource. I used the callback function to implement the custom learning rate scheduler. This improved the models performance a lot.

For training with 512 image input size and both training and validation data as input, the accuracy of the model was **1.0** with loss of **0.0018**. Fig 7 shows the loss of both model and validation at every epoch, for the initial train.

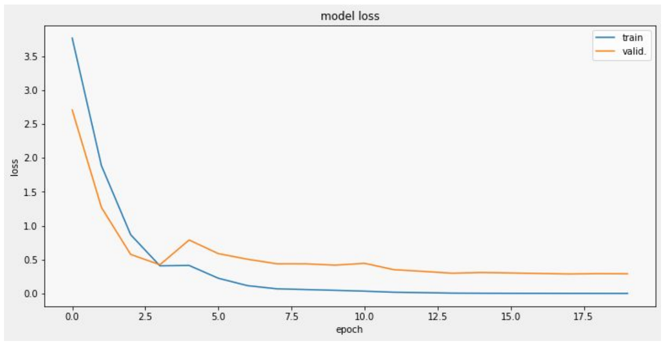


Fig. 7. Loss in training and validation with respect to epochs, for Dense Net 201 model.

2. EfficientNet B7

The EfficientNet B7 uses a model scaling method that uses a highly effective compound coefficient to scale up CNNs in a more structured manner. In this work, I used Keras implementation of EfficientNet with weights pretrained on "noisy student". The "Noisy Student" achieved 88.4 percent top-1 accuracy on ImageNet. Using the "Noisy student" method improved the classification accuracy of the model. Rest of the model configurations were similar to the DenseNet-201 model.

For final training with 512 image input size and both training and validation data as input, the accuracy of the model was **0.9971** with loss of **0.0104**.

For initial training with 224 image input size and only training data as input, the accuracy of the model was **0.9976** with loss of **0.0209**. Fig 8 shows the loss of both model and validation at every epoch, for the initial train.

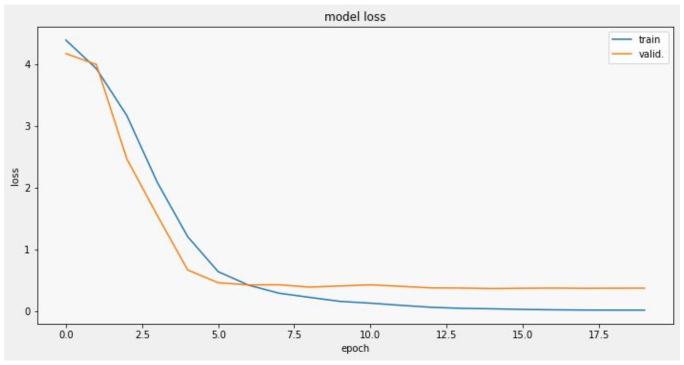


Fig. 8. Loss in training and validation with respect to epochs, forEfficient Net B7.

3. Inception ResNet V2

The Inception ResNet V2 is a hybrid Inception module, inspired by ResNet. Inception Modules are used in Convolutional Neural Networks to allow for more efficient computation and deeper Networks through dimensionality reduction with stacked 1×1 convolutions. The modules were designed to solve the problem of computational expense, as well as overfitting, among other issues.To implement this model, I used Keras Applications that contains the InceptionResNetV2 models, with weights pre-trained on ImageNet. The configuration of this model is similar to the one described in the DenseNET model.

The Inception ResNet was only trained initially since it did not perform as well as others on validation. For initial training with 224 image input size and only training data as input, the accuracy of the model was 1.0000 with loss of 0.0023. Fig 9 shows the loss of both model and validation at every epoch, for the initial train.

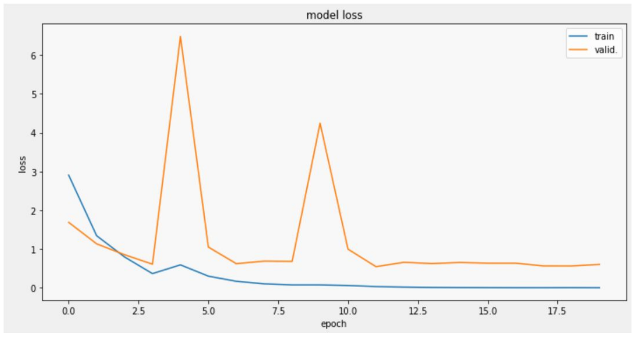


Fig. 9. Loss in training and validation with respect to epochs, for Inception ResNet V2.

These models were evaluated against the validation dataset. Fig 10. shows the validation accuracy and loss of these models.

Full Training Models	Training		Validation	
	Loss	Accuracy	Loss	Accuracy
Inception ResNet V2	0.0023	1	0.6043	0.8879
DesNet	0.0018	1	0.2905	**0.9375**
Efficient Net	0.0209	0.9976	0.3762	**0.9353**

Fig. 10. Training and validation results.

F. *ENSEMBLE LEARNING*

In layman terms ensemble learning allows us to combine the decisions from multiple models. I combined the above described models to form four ensemble models.

1. DenseNET 201, Efficient NET B7
2. Inception ResNet V2, Efficient NET B7
3. DenseNET 201, Inception ResNet V2
4. Inception ResNet V2, DenseNET 201, Efficient NET B7

There are different ensemble techniques to combine the decisions from multiple models to improve the overall performance. To find the the weighted average of the predictions for these model I used the following equations:

For pair ensemble:

$$Prediction = (alpha) * m1 + (1 - alpha) * m2$$

For ensemble of three models:

$$Predict = alpha * m1 + (beta) * m2 + (1 - alpha - beta) * m3$$

I feed this equation in the loop to find the best values of alpha and beta. The best alpha and best beta was determined on the basis of f1 evaluation score. The function applies the different values for alpha and beta and outputs the weights that gives the best f1 score.

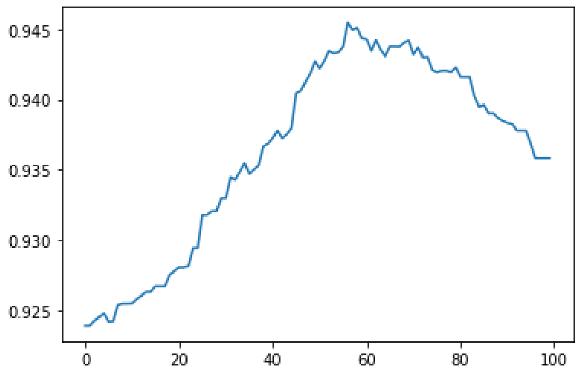


Fig. 11. F1 scores across alpha value ranging from 0 to 100. The peak shows the best alpha value at 56.

After finding the best alpha and best beta values, I used these weights to combine each of the four ensemble models. I evaluated these models against validation data. Fig 12 shows the evaluation results of these models.

MODELS	F1 score
DenseNET 201, Efficient NET B7	**0.944**
Inception ResNet V2, Efficient NET B7	0.93
DenseNET 201, Inception ResNet V2	0.933
Inception ResNet V2, DenseNET 201, Efficient NET B7	**0.943**

Fig. 12. Evaluation result of ensemble models

Since the ensemble of DenseNET 201, Efficient NET B7 and the ensemble of Inception ResNet V2, DenseNET 201, Efficient NET B7 performed the best, I decided to use them for final submission. Fig 13, 14 shows the weights used to combine these models.

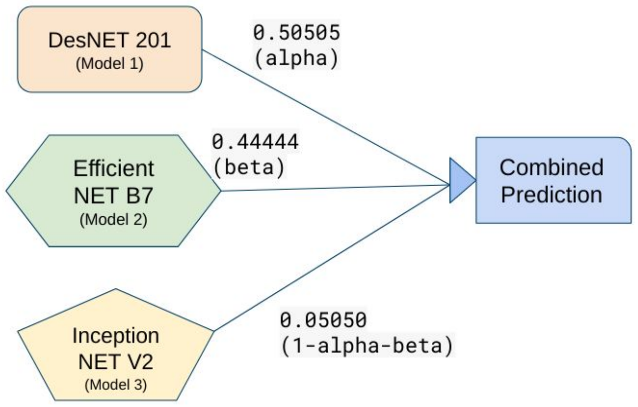


Fig. 13. Flowchart depicting the weights used to combine the predictions from three different model

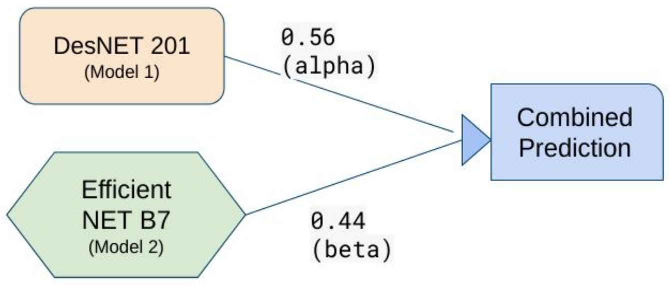


Fig. 14. flowchart depicting the weights used to combine the predictions from two different model

G. *PREDICTIONS*

The final prediction was made on the test dataset. First, I used the TTA technique to get the best prediction from each model. Second, I combined these test predictions by using the previously extracted weights.

Test-Time Augmentation (TTA) is a technique that creates multiple augmented copies of each image in the test set, having the model make a prediction for each, then returning an ensemble of those predictions. I used 10 multiples of each image and set the verbosity at 1, while predicting the test image.

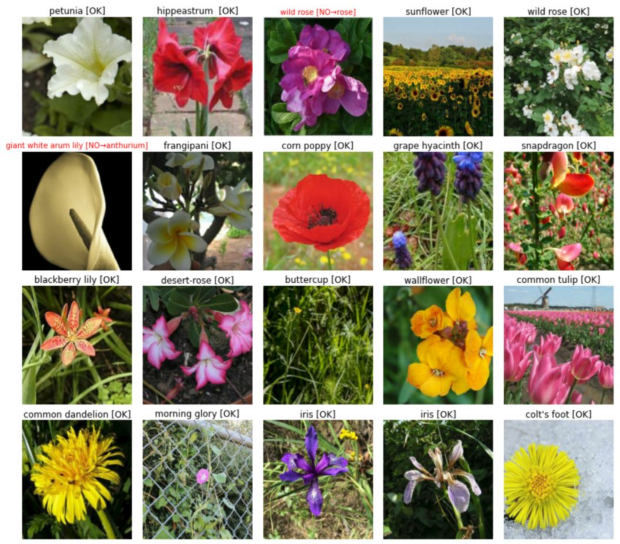


Fig. 15. Visualization of prediction done on validation set using 3 Model Ensemble.

III. RESULTS

After combining the predictions of individual models with their associated weights, I received the final predictions. Fig 16 shows the end results of two submissions I made to kaggle using two different ensemble models.

Model	Score	Rank
1. DenseNET 201, Efficient NET B7	**0.96464**	**199**
2. Inception ResNet V2, DenseNET 201, Efficient NET B7	0.93844	531

Fig. 16. Final submission score and rank on kaggle as of May 4th, 2020.

As shown in Fig. 16 the ensemble of DenseNET 201, Efficient NET B7 performed the best with an f1 score of **0.96464**. while the ensemble of Inception ResNet V2, DenseNET 201, Efficient NET B7 received the f1 score of **0.93844.**

However, It is worth noting that the 3 model ensemble was trained on input image size of 224 due to resource limitations. Also the model was trained on the training dataset while the best performing ensemble of two models was trained on both training and validation dataset.

IV. CONCLUSION

In this work, I evaluated the various ensembles of the best performing models. I used techniques like Data Augmentation to enhance the input dataset, custom LR scheduler to enhance the optimization, and TTA to enhance the predictions. The predictions of the ensemble of DenseNET 201, Efficient NET B7 performed the best with an f1 score of **0.96464**.

This project presents a deep learning approach to classify the 104 species of flowers.The most challenging part of the project was computation limitations. Managing the experiment within these resources without exhausting them was challenging.

The project provides a good exposure about the use of TPU to perform these high computation classifications. It also provides the exposure to state of the art deep learning Models. I learned about the different techniques associated with deep learning and image classification. For Future work, It will be interesting to see the addition of the attention layer in the neural network and how it affects the accuracy of these models.