Micah Simmerman
Artificial Intelligence
CSPB-3202, HW5

Designing an EfficientNetB7 Convolutional Neural Network (CNN) for Binary Classification of Histopathology Slide Samples for Cancer Detection

GitHub Repository: https://github.com/Micah614/CNN_Classifier_Cancer_Detection

Introduction

The purpose of this Kaggle coding competition is to develop and train a convolutional neural network (CNN) based, binary image classifier capable of accurately identifying cancer tissues in histopathology slides of tumor biopsies. The EfficientNetB7 CNN model system was selected for this purpose based on a a seminal article written by Mingxing Tan and Quoc V. Le and published in arXiv on September 11th, 2020.

The generation and training of the EfficientNetB7 CNN developed in this report required the use of a ETL (extract-transform-load) data pipeline to separate the binary classes presented in the "train" folder while transforming their file encoding to ensure forward compatibility with keras preprocessing libraries. Model weights were adjusted twice, and trained each time for a period of 20 epochs. The final EffficientNetB7 CNN demonstrates a validation accuracy of 90.91 percent, a surprising number, given the suggested training period of 90 epochs required to ensure optimality. The model boasts some 802 hidden layers, including 55 trainable layers, and a two-stage relu-flatten-sigmoid activation function.

The model was fed using a custom-made ETL data pipeline specifically designed to extract, transform, and load the Kaggle "Histopathologic Cancer Detection" dataset for subsequent training and validation. This pipeline successfully constructed a png-transformed purified directory folder that was used to train and validate the EfficientNetB7 CNN network model. Numerous GitHub repositories and online tutorials were referenced throughout the course of this project, including the Keras and TensorFlow official documentation libraries (see https://www.tensorflow.org/tutorials/keras/save_and_load).

Dataset Directories and the ETL Pipeline

The original dataset folder was collected from https://www.kaggle.com/competitions/histopathologic-cancer-detection. The .tif-encoded image file folder is composed of "test" and "train" subdirectories, which contain 220,025 class-labeled training images and 57,458 unlabeled test images, respectively. All images included in the primary file are 96x96 pixels in size and needed to be encoded in an acceptable file format for the kera preprocessing tool libraries that are used to train the neural network.

The ETL data pipeline was used to duplicate the "test" and "train" image folders into a .png formatted directory file, referred to throughout this document at "PURE_DIRECTORY". The training image files were copied from the original folder and divided into two PURE_DIRECTORY sub-folders based on their assigned class labels (0 for benign, 1 for metastatic). A small validation directory was constructed at the same time, and in the same manner, and composes roughly 10% of the training files that were diverted

for the purpose of model validation. The ETL file extraction was performed with the use of a Pandas data frame, which is itself composed of the “train_labels.csv”, included in the original download folder.

Training Batch Construction

The “cv2.imread” library tool was used to examine the image file data in an [RGB] numpy array format. After verifying the tif-to-png file conversion had taken place, the PURE_DIRECTORY/training and PURE_DIRECTORY/validation directories were loaded into a Keras-style batch generators (i.e., iterator objects). These TF batch iterators were constructed using “tf.keras.utils.image_dataset_from_directory”, and verified the presence of 198,022 files in “PURE_DIRECTORY/training” and 22003 images in “PURE_DIRECTORY/training”, each belonging to 2 classes.

Constructing the EfficientNetB7 CNN Model

The EfficientNetB7 CNN model was selected for this project based on a recent research article by Mingxing Tan and Quoc V. Le, in which the benefits of *compound scaling* in convolutional neural networks is explained. Accross the eight EfficientNet models described and compared by this article, EfficientNetB7 stands out as a clear winner in terms of its’ accuracy and resource-balance efficiency.

Tan and Le’s “compound scaling” approach works a bit like a compound telescope, in that the structure can be adjusted by scaling structural features that share a correlated effect on the model’s performance. According the Tan and Le, these dimensions are the CNN’s “depth”, “width”, and “(image) resolution”. The act of balancing these parameters in unison is what is meant by the term “compound scaling”, and the technique does indeed appear to confer many advantages in terms of model accuracy, and sharp decreases in the number of parameters required to train the neural network to rival a competing neural network system. (please visit https://arxiv.org/abs/1905.11946 to download the article).

```
# CELL 12
base_path = "C:/Users/jmica/Desktop/histopathologic-cancer-detection/PURE_DIRECTORY\\training_directory\\"
TRAINING_LOGS_FILE = "training_logs.csv"
MODEL_SUMMARY_FILE = "model_summary.txt"
# MODEL_FILE = "benign_v_malignant_pretrained.h5"
#  https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142
# https://keras.io/api/metrics/

model = Sequential()
efficient_netB7 = efn.EfficientNetB7(weights='imagenet', include_top=False, input_shape=input_shape)
for index, layer in enumerate(efficient_netB7.layers):
    if index < 748:  # 761
        layer.trainable = False
    print(index)
    print(layer)

model.add(efficient_netB7)
model.add(Dense(1024, activation='relu'))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid')) #, name="output"
model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(learning_rate=0.0001),
              metrics=['accuracy'])

with open(MODEL_SUMMARY_FILE,"w") as fh:
    model.summary(print_fn=lambda line: fh.write(line + "\n"))
```

Figure 1.) EfficientNetB7 model parameters comprising the binary image classifier’s neural network structure.

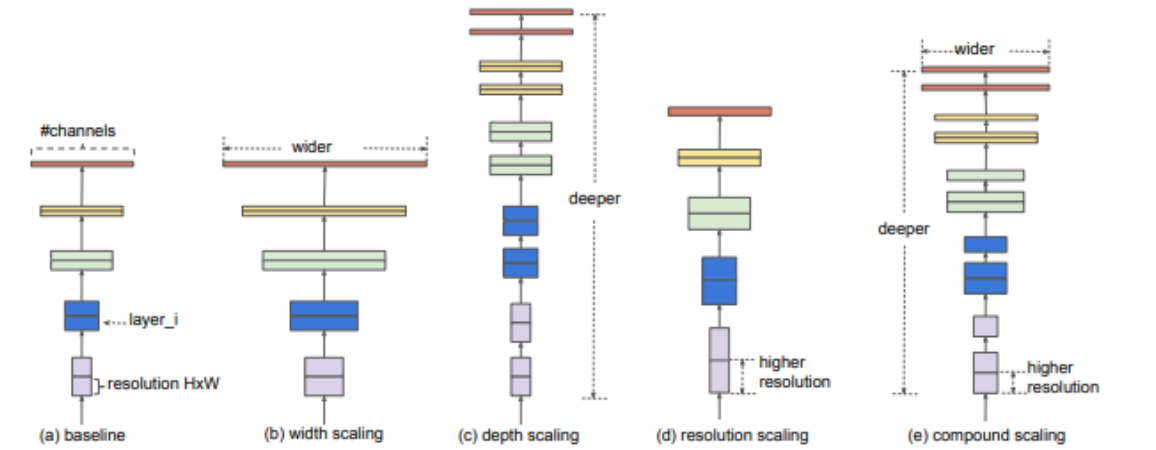


Figure 2.) A visual depiction of the meaning of the term "compound scaling" according to Tan and Le's article: "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", arXiv: 1905.11946, 11Sep2020.

Adjusting the Parameters of EfficientNetB7

The EfficientNetB7 CNN consists of 805 network layers that can be controlled by modifying the operational roles of each layer hidden inside the neural network. In particular, EfficientNetB7 model layers possess a Boolean "trainable" class variable, which controls whether a particular network layer contains trainable (hyper)parameters or not. Activating a layer will assigns it to the task of feature extraction, while assigning the layer as non-trainable designates it as a focusing convolutional layer with the primary task of increasing the resolution of captured image features.

Note: Please see the file "model_layer_details" located in this project's GitHub repository for more information about the convolutional layer features comprising this binary classifier's neural network system.

EfficientNetB7 contains the following class member arguments, returning a model instance (source: https://keras.io/api/applications/efficientnet/):

- **include_top:** whether or not to include fully-connected (ImageNet) pretrained layers
- weights: "imagenet" by default. Developers can also choose use their own.
- **input_tensor:** optional keras tensor to use as image input for the model
- **input_shape:** optional image shape tuple with 3 input channels, must be included when include_top=False
- **pooling:** 'None', 'avg', or 'max'
- **classes:** optional number of classes the images will be assigned to
- **classfier_activation:** the activation function to use in the top layer, ignored unless include_top=True

Programmers are free to adjust their model by selecting an alternative optimization methods (e.g., AdaDelta, or RMSProp). Decision layers can also be added to the model by invoking "model.add" (see

https://keras.io/guides/sequential_model/ for more information). Figure 3 presents a concise summary of the Gen-1 image classifier model.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnet-b7 (Functiona  (None, 3, 3, 2560)        64097680
 l)

 dense (Dense)               (None, 3, 3, 1024)        2622464

 flatten (Flatten)           (None, 9216)              0

 dense_1 (Dense)             (None, 1)                 9217

=================================================================
Total params: 66729361 (254.55 MB)
Trainable params: 20407521 (77.85 MB)
Non-trainable params: 46321840 (176.70 MB)
_________________________________________________________________
```

Figure 3.) Model summary of the EfficientNetB7 CNN Cancer Image Classifier model, training session #1.

Model Training and Validation Results

The first round of model training and validation was performed using 761 inactive layers, trained over 20 epochs. The training and validation session took over 17 hours and demonstrated considerable variance in its' validation accuracy. By the end of the training session the model predictive validation settled at around a 87% accuracy score

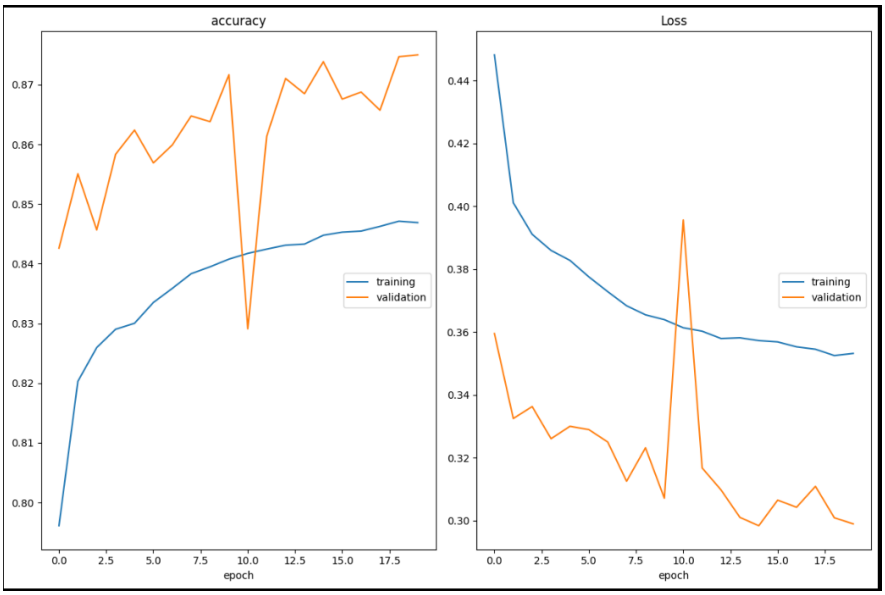


Figure 4.) Left: training and validation accuracy curves, collected over 20 epochs of network training using the CNN parameters described by Figure 2. Right: binary_crossentropy loss function for training and validation runs.

Unfortunately, the file path ('MODEL_FILE') where the model's training weights were intended to be stored had not been assigned prior to the initial training/validation session. This mistake necessitated a complete retraining of the network model, but provided a meaningful opportunity to reorient the model.

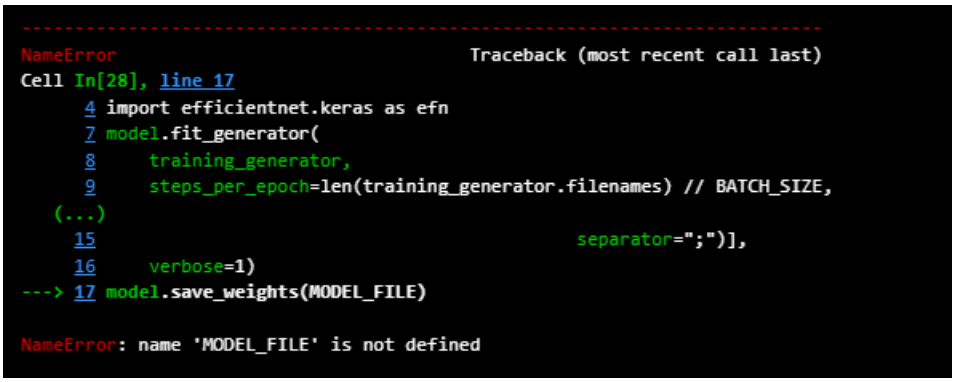


Figure 5.) An unfortunate set-back in EfficientNetB7 binary classifier training that led to the readjustment of active network layers and increased batch size to boost the CNN model's performance.

Several short (i.e., 1-2 epoch) training sessions were conducted to isolate parameter adjustments that could result in faster model convergence and reduced training curve variance. It was discovered after a short time that doubling the batch size resulted in sizable decreases in model variance. It was also noted that increasing the number of active (i.e., trainable) layers appears to improve the binary classifier's accuracy score. A sizeable decrease in model training time was also observed, requiring only 15 hours instead of 17.

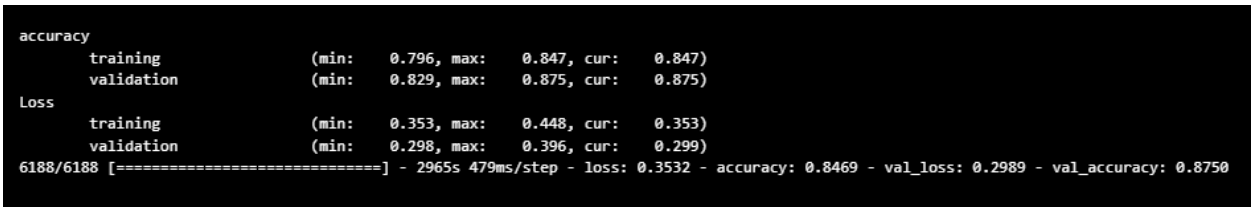


Figure 6.) Final summary of the first 20-epoch length training session that was discarded because of a missing file reference encountered as a late runtime error.

EfficientNetB7 Classifier – Training and Validation (Attempt #2)

The newly-adjusted model was trained for 20 epochs using a batch size of 64 (instead of 32). The network active layers were set to begin at 748 instead of 761, introducing an additional 13 layers that seemed to improve its' accuracy. These adjustments appear to have noteworthy effects on the model's training and validation accuracy scores. The new model also appears to converge upon a steady state faster than the previous model, though it is not clear which change is responsible for this effect. The improved training rate is perhaps not all that surprising, given the inverse relationship between batch size and the number of epoch steps required to complete a training session.

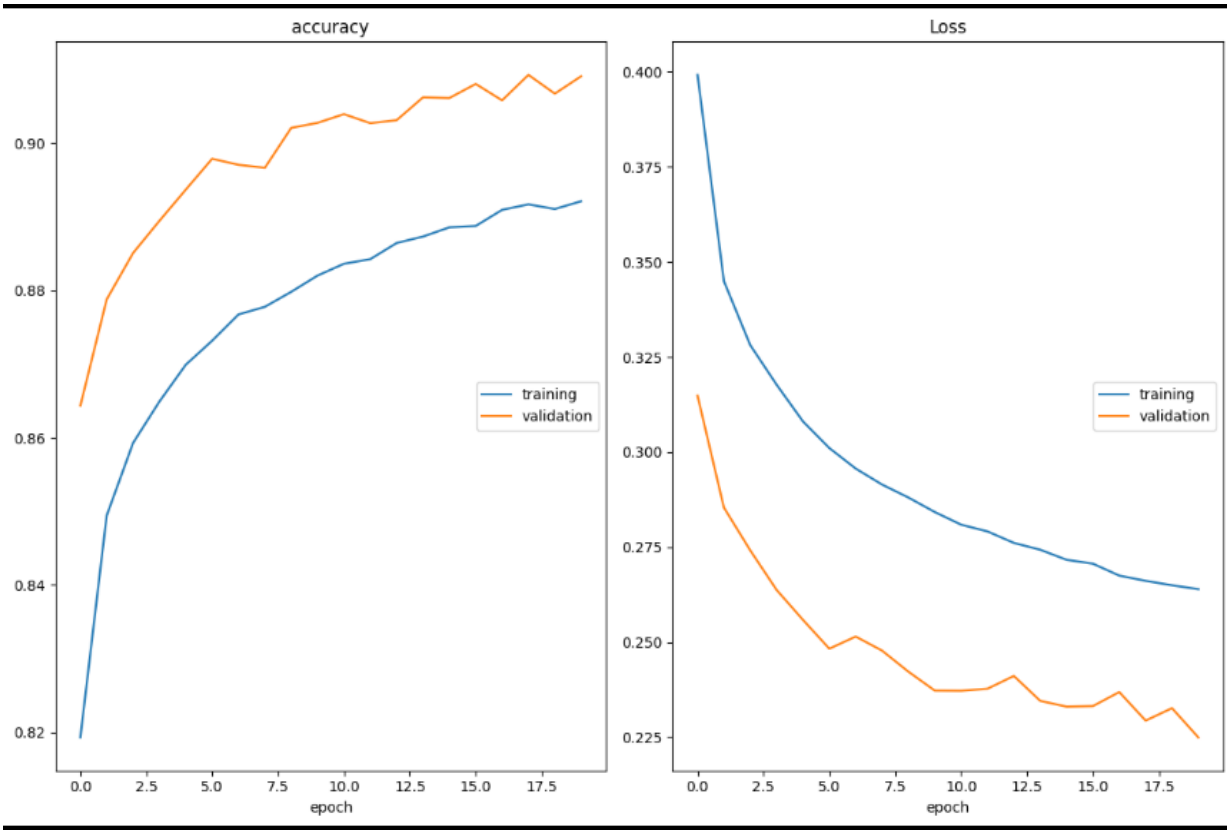


Figure 7.) New metric curves for the modified EfficientNetB7 CNN classifier model. Notice the increased accuracy and diminished variance associated with these curves when compared to previous 20 epoch training session.

```
accuracy
        training                (min:    0.819, max:    0.892, cur:    0.892)
        validation              (min:    0.864, max:    0.909, cur:    0.909)
Loss
        training                (min:    0.264, max:    0.399, cur:    0.264)
        validation              (min:    0.225, max:    0.315, cur:    0.225)
3094/3094 [==============================] - 2709s 875ms/step - loss: 0.2639 - accuracy: 0.8921 - val_loss: 0.2249 - val_accuracy: 0.9091
```

Figure 8.) The accuracy and Loss metric table for the Gen-2 classifier system, trained and validated in the same manner as the previous model

Generation of the Kaggle Submission File

A .csv formatted output file was created to collect the CNN classifier-assigned labels for every sample in the ‘test’ folder. This file was generated by composing an intermediate data frame using Pandas. The data frame was built by iterating through “PURE_DIRECTORY//test” to collect the id of every image in that folder and load it into the frame, while stripping off the “.png” file extensions and initiating “label” values to zero.

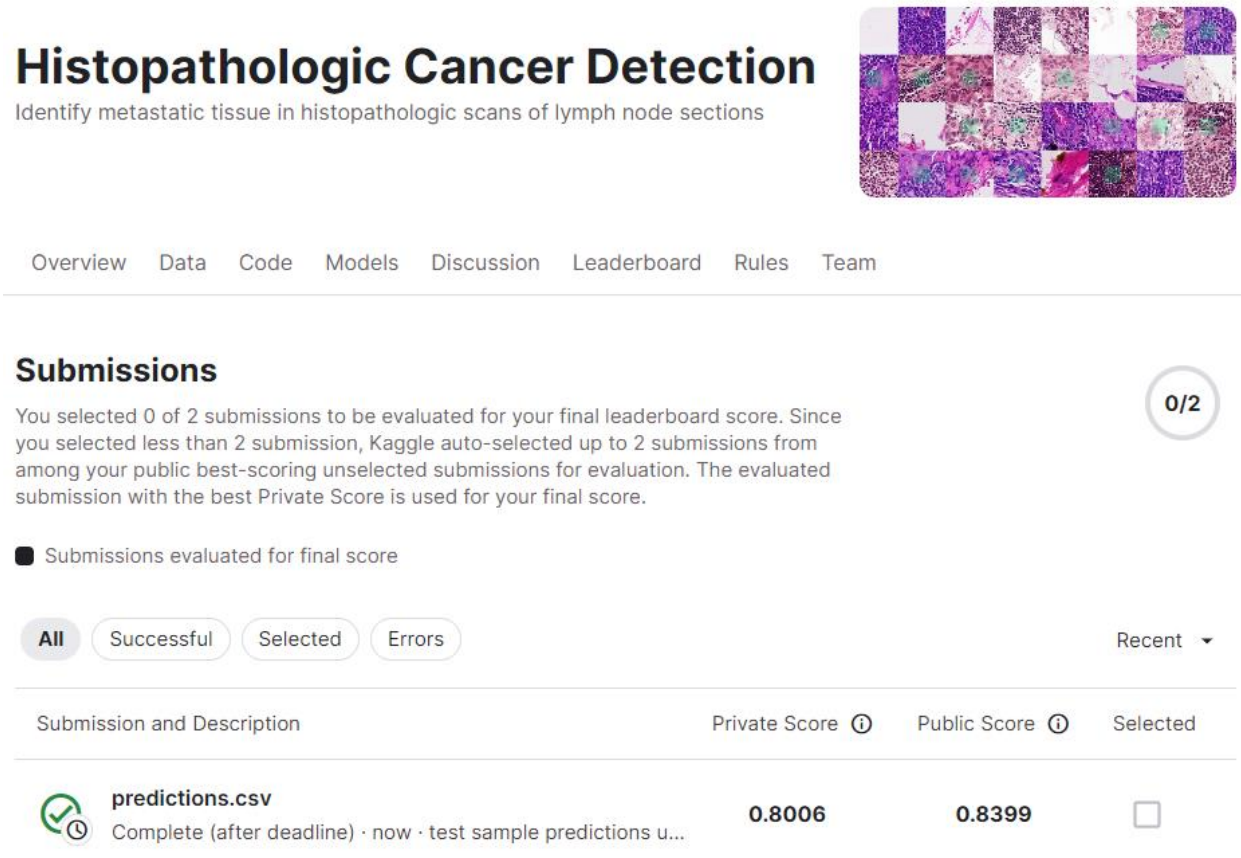
This data frame object was then used like an index to iterate through “PURE_DIRECTORY//test” once more, this time converting the image data and feeding it to the generator. On each iteration, the predicted label was used to overwrite the ‘id’ value corresponding to each sample. The finalized data frame was then written to a csv format for final submission.

Conclusion/Discussion

One subject that may be worthwhile to investigating is the impact that a well-abstracted GPU could have on the performance of the EfficientNetB7 system. This aspect of TensorFlow was not regrettably not explored during the course of this assignment, owing mainly to an acute lack of TensorFlow GPU driver support. It seems this obstacle could be overcome in the future by invoking a python virtual environment or virtual disk image, although these solutions are far from certain.

Overall, the EfficientNetB7 CNN architecture emphatically delivers on the promises outlined by Tan and Le, exhibiting a very reasonable network training rate, and simple parameter adjustment that enables rapid optimization of existing network models. Based on the experience acquired from this report, it is likely that the model may be optimized even further, perhaps breaking the 95% accuracy boundary with enough optimization and training.

Kaggle Submission Proof



Works Cited

- https://www.tensorflow.org/tutorials/keras/save_and_load
- https://www.kaggle.com/competitions/histopathologic-cancer-detection
- https://keras.io/api/applications/efficientnet/
- https://keras.io/guides/sequential_model/
- “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”; Mingxing Tan, Quoc V. Le; arXiv:1905.11946v5 [cs.LG]; 11 Sep 2020 => https://arxiv.org/abs/1905.11946