CSPB 3202 HW 5 - Kaggle Competition

Taylor Larrechea

The University Of Colorado Boulder, College Of Engineering And Applied Science

Abstract

This assignment focuses on training a Machine Learning model that can accurately classify images that represent cancer. For this model, the python package Tensorflow was used to train the model. This article will discuss the methodology used to train the model, the results of the model, and the conclusions that can be drawn from the results.

Imports

The first step that was used in creating a model to classify images was to import the necessary libraries. The libraries that were imported and used in this model were:

- **pandas**
- **tensorflow**
  - **keras.preprocessing.image** - ImageDataGenerator: A class that generates batches of tensor image data with real-time data augmentation.
  - **keras.applications** - ResNet50: A pre-trained model that can be used to classify images.
  - **keras.models** - Model: A class that groups layers into an object with training and inference features.
  - **keras.layers**
    - **GlobalAveragePooling2D**: A class that applies average pooling on the spatial dimensions until each spatial dimension is one.
    - **Dense**: A class that implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True).
    - **Input**: A class that is used to instantiate a Keras tensor.
  - **keras.callbacks**:
    - **ModelCheckpoint**: A class that saves the model after every epoch.

Once these models were imported, the model was now set to be trained.

Model Training Algorithm

To begin training this model, the first step was to get the data from the CSV file that was provided by Kaggle.

Data Preprocessing

The name of the CSV file that was provided was *train_labels.csv*. This file was loaded into a pandas DataFrame where label was loaded into one column and the image ID was loaded into another column.

This data was then split into a training and validation set. The training set was 80% of the data and the validation set was 20% of the data. The intention of using a validation set was an attempt to prevent overfitting of the model.

Data Augmentation

The next step was to then augment the data that was loaded. This was done using the *ImageDataGenerator* class from the *keras.preprocessing.image* module. This class was used to generate batches of tensor image data with real-time data augmentation. The data augmentation that was used was to rescale the images by 1/255, rotate the images by 20 degrees, zoom the images by 20%, and flip the images horizontally.

Model Creation

After the data was processed and then subsequently augmented, the next step was to create the model. The model that was used was the ResNet50 model from the *keras.applications* module. This model was used because it is a pre-trained model that can be used to classify images. The model was then set to be trained on the training data. This model was then trained with a total of 10 epochs. The results of each subsequent epoch were saved using the *ModelCheckpoint* class from the *keras.callbacks* module. This model was then saved using the *model.save()* method.