UNITED STATES MILITARY ACADEMY

HW2

CS485: SPECIAL TOPICS IN COMPUTER SCIENCE

SECTION G2

MAJ DANIEL RUIZ

By

CADET JACK SUMMERS ’22, CO A4

WEST POINT, NEW YORK

1 MARCH 2022

_J.S__ MY DOCUMENTATION IDENTIFIES ALL SOURCES USED AND ASSISTANCE RECEIVED IN COMPLETING THIS ASSIGNMENT.

___ _ I DID NOT USE ANY SOURCES OR ASSISTANCE REQUIRING DOCUMENTATION IN COMPLETING THIS ASSIGNMENT.

SIGNATURE:______________________________

1. The path that took me to my final model started with data analysis. I first figured out how to download the data, load it, and visualize it. Next, I created the same callbacks I used for HW1 because I wanted to something to steer my algorithm in the right direction. Due to the fact I knew the data set was images; I knew I wanted to use a data generator to augment that data in order to artificially expand the data set size. I could not figure out a way to feed separate validation and training data into the ImageDataGenerator so I used the validation_split command. In turn, I combined all of the validation and training data into one numpy array and fed it into the generator. Due to the fact the data set is images, I used a convolutional neural network. I first added Conv2D layers, the first layer with size 28 x 28 x 1 because the images are grayscale 28 x 28. I also added maxPooling2D layers to reduce the number of feature-map coefficients to process, as well as to induce spatial-filter hierarchies.

   Next, I flattened the output of the last MaxPooling layer to input it into the dense layers. I added dense layers with large amounts of neurons because when I was training the model, I was not overfitting, just underperforming. When compiling the data, I used categorical crossentropy because the problem is multiclass in nature. Finally, when training the model, I used 1181 steps per epoch because 151284 training images / 128 batch size = 1181.9. Training the model typically took 1.5 hours, so I saved the entire model with model.save('file.h5'). In turn, I could load the saved model later to test it and also save time. I stopped tunning hyperparameters because the model would take over an hour to train each time.

2. I did not have any problems with overfitting, my model would simply stagnate at 0.61 validation and training accuracy. I knew I was not overfitting because when I graphed the data, both the validation and training accuracies remained approximately the same. Because I did not struggle with overfitting, I did not implement countermeasures. Specifically, I did not use dropout layers or the ReduceLROnPlateau callback function.

3. Universal Machine Learning Workflow:

   Step 1: I must take a grayscale image of a kidney cells and classify it as 0 – 7 inclusive. From this I understood I must use a CNN, with an input size of 28 x 28 x 1 (grayscale). Additionally, I understood that this was a multiclass classification problem.

   Step 2: To measure success the model's accuracy and validation accuracy were plotted and compared. If both stayed approximately equal, and above 0.61 then the model would be considered a success. The model would be considered a success because there was no overfitting and it beat the competition benchmark.

   Step 3: I used a training generator that allowed me to use the validation_split_command. Twenty percent of the data was used for validation. Testing data was provided separately.

Step 4: First the data was loaded, then shaped into size (dataset size, 28, 28, 1). Then the data was fed into a training a training generator and rescaled. In turn the data was turned into vectors and normalized.

4. I used the following data augmentation techniques: rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2, horizontal_flip=True. Augmenting greatly increased the accuracy greatly by artificially expanding the data set.

5. I think the data set was not ideal for identifying kidney cells because the images were very blurry. In turn I believe the algorithm may have had a difficult time extracting feature. I believe the main feature that the model was able to extract would be the perimeter shape of the cell, for it was the most distinguishable feature.

Bibliography

[1] “numpy.load — NumPy v1.10 Manual,” *docs.scipy.org*. https://docs.scipy.org/doc/numpy-1.10.1/reference/generated/numpy.load.html (accessed Feb. 27, 2022).

[2] “numpy.append — NumPy v1.22 Manual,” *numpy.org*. https://numpy.org/doc/stable/reference/generated/numpy.append.html (accessed Feb. 27, 2022).

[3] N. Gervais, “python - ValueError: Input 0 of layer sequential is incompatible with the layer: : expected min_ndim=4, found ndim=3. Full shape received: [8, 28, 28],” *Stack Overflow*, 2021. https://stackoverflow.com/questions/63279168/valueerror-input-0-of-layer-sequential-is-incompatible-with-the-layer-expect (accessed Feb. 27, 2022).

[4] J. Brownlee, “How to Normalize, Center, and Standardize Image Pixels in Keras,” *Machine Learning Mastery*, Apr. 02, 2019. https://machinelearningmastery.com/how-to-normalize-center-and-standardize-images-with-the-imagedatagenerator-in-keras/.

[5] “Save and load models | TensorFlow Core,” *TensorFlow*. https://www.tensorflow.org/tutorials/keras/save_and_load.