Spring 2022: CSEE5590/490 – Special Topics

Python and Deep Learning Module-2 - ICP-10

Lesson Overview:
In this lesson, we are going to discuss Image classification with CNN.

Use Case Description:
Image Classification with CNN
1. Training the model
2. Evaluating the model

Programming elements:
1. About CNN
2. Hyperparameters of CNN
3. Image classification with CNN

Source Code:
Provided in your assignment folder and assignment repo.

In class programming:

1. Follow the instruction below and then report how the performance changed. (Apply all at once)

> Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
> Dropout layer at 20%.
> Convolutional layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
> Max Pool layer with size 2×2.
> Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
> Dropout layer at 20%.
> Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
> Max Pool layer with size 2×2.
> Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
> Dropout layer at 20%.
> Convolutional layer,128 feature maps with a size of 3×3 and a rectifier activation function.
> Max Pool layer with size 2×2.
> Flatten layer.
> Dropout layer at 20%.
> Fully connected layer with 1024 units and a rectifier activation function.
> Dropout layer at 20%.
> Fully connected layer with 512 units and a rectifier activation function.
> Dropout layer at 20%.
> Fully connected output layer with 10 units and a softmax activation function

2. Change the previous model into Keras Functional API model.

2.1 Apply the following callbacks to the model:

- ModelCheckPoint.

- ReduceLROnPlateau.

- EarlyStopping, use the “restore_best_weights” parameter.

3. Save the model. (Store this model will be required for future ICP).

4. Predict the first 4 images of the test data. Then, print the actual label for those 4 images (label means the probability associated with them) to check if the model predicted correctly or not.

5. Build your own dataset by collecting images from the internet, for example:

- Transportation images (Airplanes, Trains, Cars, ..)
- Animals (Cats, Dogs, ..)
- (You can use your project dataset).

5.1 Train the model on your dataset and report the accuracy and type of pre-processing that needed to be done.

5.2 Plot the training and validation accuracy.

5.3 Save the model as a file and load it again to predict on unseen images (test data).

** Follow the IPC rubric guidelines.

Submission Guidelines:

1. Once finished document your code and make sure all parts of the assignments are completed.
2. Push your code to your GitHub repo and update the ReadMe file, add your info, and partner info.
3. Submit the assignment on Canvas.
4. Present your work to TA during class time to prove the execution and complete submission.

After class submission:

1. Once finished document your code and make sure all parts of the assignments are completed.
2. Push your code to your GitHub repo and update the ReadMe file, add your info, and partner info.
3. Submit the assignment on Canvas before the deadline.
4. Record a short video (3~7) minute, proof of execution and complete assignment.
5. Add video link to ReadMe file.

Note: *Cheating, plagiarism, disruptive behavior, and other forms of unacceptable conduct are subject to strong sanctions in accordance with university policy. See detailed description of university policy at the following URL: https://catalog.umkc.edu/special-notices/academic-honesty/*