Convolutional Neuron Network By JUSTINE Jean using Chatgpt

I. Convolution

The first task was to apply Convolution to some images and show the results, for this task I mainly use the code available in hands on machine learning chapter 14, all references at the end.

Firstly, we used sklearn to load some images, and scale the images, create filters using numpy and then we create the convolution layer using tensorflow: tf.keras.layers.Conv2D. The kernel slide over the image with a matrix and perform calculation to re calculate according to the pixel in the kernel and add it to a feature map.

Using the kernel previously created, we apply this convolution layer to the images creating a feature map then we showed the results using matplotlib. We did that on two different images, using different kernel, for vertical and horizontal convolution.

Then we apply pooling, we used the same method as convolution but we used this function to create the pooling: max_pool = keras.layers.MaxPool2D(pool_size=2), for max pooling, this function reduce the spatial size of the feature map by dividing it into region for same size(here 2*2) and taking the maximum value in each region and avg_pool = keras.layers.AvgPool2D(pool_size=2) for average pooling, it work like max pooling but took the average, and finally, for depth wise pooling we created the function ourselves, and apply max pooling across the channel of the image instead than spatially like before .

II. Convolution Network for Mnist Dataset

We created a Convolution Network, to train on the Fashion MNIST. To do so, we first split the dataset between train validation and test set from the fashion MNSIT set we imported using keras.

Then, we create the model using the function Sequential that allow us to add layer to our model. Those layers are create using the function partial from keras, that allow us to create layers using the convolution algorithm with: keras.layers.Conv2D, then we create the model with sequential, using a mix of multiple layer like maxpooling2D, dropout, Dense and convolution of course. Then we compile the model with accuracy as metrics, nadam as optimizer and sparse_categorical_crossentropy, as loss function, we used 10 epochs, and the validation set as the validation data. Then we train it, save it and evaluate it, the training took 67 minutes, we got an accuracy of 89%, compare to the neural network we created in the previous assignment, in which we had 0.1 accuracy at most, is very good, but to put things in context the NN we created consisted of 100 of the same type of layers and no droupot or any tuning, to try different optimizer. So the results are to be put back in context.

III. ResNet-34

A ResNet-34 is a model with an architecture like that :

Initial Layers:

- A 7x7 convolution with stride 2.
- Batch normalization, ReLU activation, and max pooling.

Residual Blocks :

- 3 blocks with 64 filters.
- 4 blocks with 128 filters.
- 6 blocks with 256 filters.
- 3 blocks with 512 filters.
- Each block contains two convolutional layers.

Global Average Pooling:

- Reduces the spatial dimensions of the feature map.

Fully Connected Layer :

- Outputs class probabilities.

So this is exactly the structure of our model. To do so we used a function that we create called Residual Units to create our residual blocks. That are different than a normal nn, it can skip connection. We also still use the function previsouly describe: DefaultConv2D that we created to create the convolution layers. Finally, we printed our model summary to see its architecture and parameters.

IV. Xception Model for Transfer learning

I had a lot of struggle with this task because tnesorflow_datasets weren't working for me and I could not import the dataset so after trying to resolve this problem, I resorted myself to just download it manually and then split it manually into test, train and validation set.
I asked chatGPT to help me on that as I was having trouble accessing and modifying the data. So after finally splitting it, I normalized it, and resized it to 299*299.
After that we finally could create the model and train it, I once again used code from Hands on Machine Learning that was provided with the assignment.
So we created the keras model, with xception as the base model, we use global average pooling, and as an output layer,

we used 128 units and softmax for activation and that’s our model.
Then we needed to train it, we use the fit function like usual, with SGD as optimizer with 0.2 learning rate, 0.9 momentum, and 0.01 decay. We trained it using train set as the training data, and of course val set as the validation data, 5 epochs and finally, we evaluated it and got as accuracy: 0.2 which was weak but during the training we got an accuracy of 0.6 so an overfitting is happening.

V. Conclusion

To conclude this assignment, we learned how to use pooling, and convolution to work with dataset of images, we learned what each of those function do, how to use them, how efficient they were use, we also learned about transfer learning, the ResNet-34 CNN, we could compare it with other model of NN.

It was a short assignment and a short report thanks to: https://github.com/ageron/handson-ml2/blob/master/14_deep_computer_vision_with_cnns.ipynb

That helped me a lot, and all of my code is based on that if it’s not directly it, because this chapter answered most of our questions for this assignment. I used chatgpt to help me explain me everything when I needed it or when I did not understand it, I alos used him for the final task as I said because I could not import the dataset and was struggling a lot with it, finally I also used it to help me when code’s error occurs, and I didn’t have a clue on how to solve it. I will say it once again but Hands on Machine Learning did most of the job, and I could learn very efficiently thanks to that as it was easy to understand and precise compare to our tasks.