CS604 – Deep Learning for Computer Vision

Assignment 2:

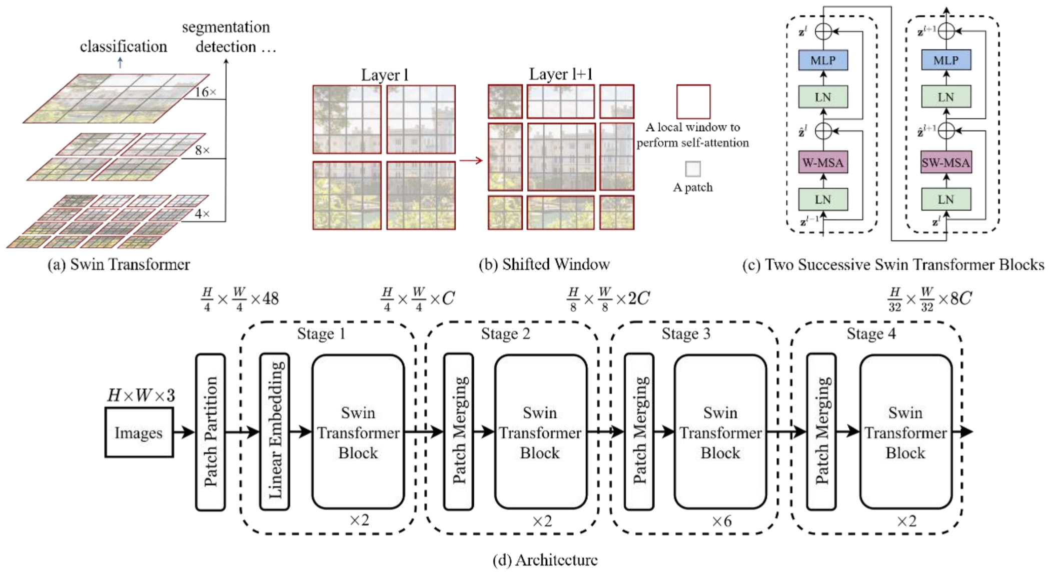
Image Classification - Report

Noah Kuntner

Description:

For the assignment, I have used the recent Swin Transformer as the main pre-trained model. Hereby given the time constraints I ended up using the 224x224 swin_base_224-model. Swin is a vision transformer that builds hierarchical feature maps by merging patches in deeper layers and uses a single linear computation complexity to input each image size due to the computation of self-attention only within each local window. The swin_base_224 has approximately 88 million parameters to train.

Initially, I have started to train on my local machine with other models, however, due to the sheer computational size needed I have switched followingly to Colab Pro, before finally resorting to the Kaggle kernel, which provided me with the fastest computational time.



SwinTransformer – General Architecture

Structure of the model:

In my model, I have used a diverse set of inputs, as I have tried to take in diverse values for the k-folds, the inputs of ImageDataGenerator(), and the number of epochs running on model.fit(). I have used for my final submission a batch_size of 16, 5 k-folds, and 15 epochs.

Firstly, to reduce CPU usage and provide a better more balanced validation set, as well as address the issue of having problems with Colab timing out due to the sheer amount of Ram-usage needed to load the data into the model and run it, (it used to crash after 2-3 epochs maximum), I have used k-folds to alleviate the stress and reduce the computation needed to load the data into the model, as well as providing more stability in regards to the validity of the validation set.

Followingly, to augment the data being loaded in the model, I have used ImageDataGenerator() to generate different types of data for the set, while implementing horizontal flip, vertical flip, rotation range, width shift, heigh shift and fill mode and also for preprocessing the efficientnet.preprocess_input method.

For the creation of the model, I have used an input size of 224x224 for the image size of the Swin-base-224-model, with an additional resizing layer. I have found to obtain optimal results when setting the dropout equal to 0.5 and adding an additional dense layer with 75 units and using 'softmax' as the activation function on the output layer.

To compile the model, I have used 'sparse_categorical_crossentropy' and the categorical accuracy was calculated, as well as for the top 5 most accurate classes chosen.

Furthermore, for the optimizer, I have tested the stochastic gradient descent, RMSProp, and Adam, but I have found that in my case for this assignment, to have achieved with the base Adam. In my other models in prior use, I was using the stochastic gradient descent with a learning_rate of 0.008 and a momentum of 0.9, but with Swin case Adam performed better for me.

I have run the training for my models for 20 epochs, as my model seemingly started to converge after about a half a dozen epochs. Furthermore, I have resorted to not unfreezing the last few layers, as I was receiving a worse result in my past experiments, as I was dealing with problems in regard to the batch normalization. Thus, I still need to optimize this part of the model.

The model could still be improved significantly, however, due to the time constraints of not having been able to run more extensive experiments on Colab Pro, where I do not risk the model being timed out, I will only do so after the submission.

Hereby, we may point to the fact, that the test set is very different than the train set, e.g. pictures were taken at a different time of day.

Final Notes:

I have tried implementing other models such a EfficientNetV2-B3, however, I had a hard time getting my model up and running efficiently in EfficientNetV2 in conjunction with tf.hub, as my scores were unfortunately very modest at best. Also, it has taken me a lot of time to get the functional-api of Sequential() to work with newer models, which cost me a lot of time.

Initially ,I wanted to implement an ensemble model between Swin and EfficientNetV2, however, I have been unable to do so in the time provided. I feel like I was still able to learn a lot of the ins-and-outs of Tensorflow and I will be more prepared for the next challenge, aiming to build the aforementioned ensemble-model with EfficientNetv2 and Swin-Transformer. Now I feel also way more confident about the Sequential-API and will try to compete in other Kaggle-challenges soon.