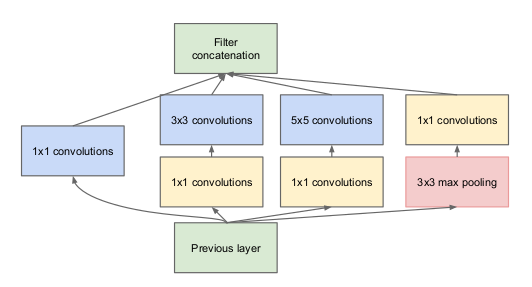
Choose a dataset and a model s.t the model performance is decent after a few epochs.

Train the model on this dataset using different combinations of hyperparameters, initial weights, and order of data loader. Say we get different models, m1, m2, m3, …m10 (inception shapes) using different combinations of their conditions, make sure they all perform decent. Then do layer feature attribution on all of them, and observe the difference between these attribution scores on each layer in different models.

Say we look at layer1, l1, m1 to m10 trained will have different attribution on l1. How large the difference is, and which condition seems to create more variance on these differences?

If the rank of attribution on each layer hasn’t even changed much, then we might say that the important neurons in this model structure are stable. Then we shall observe whether this situation can be generalized. If it’s true that models for certain datasets always have reliance preference, what observation can we make, and under what conditions are the preferences similar?

1. Before comparing rank, we can take the average attribution on each layer or kernel based on the number of outputs it generates before concatenation or weighs number, to have a better look at the most important feature in this model for this dataset, ignore the layers other than inception kernels. (but remember the key is not finding the most responsible kernel, but comparing the difference between importance later, so it doesn’t matter that much)
2. Make sure that the inception kernel used doesn’t depend on other kernels in this stack. For example, for the shape below, do not use the two 1\*1 below. For a fairer comparison, don’t use pooling before one particular kernel, use it then input the result to all kernels.



If the rank of attribution on each layer changed a lot, then we might say that the important neurons in this model structure are not stable, here’s a conclusion: for this model, this dataset, the training process doesn’t have any preference on its reliance on certain layers.

If ranks are not easy to compare in a fair way (but remember the key is not finding the most responsible kernel, but comparing the difference between importance), then just observe the difference in attribution scores for a layer between each training. See if they are stable or not.

Question: what to compare with, how to decide whether a change is enormous?

Possible baselines:

Two trained models, using a different order of data loader iterations

Two trained models, using different initial weights

Two trained models, using different other hyperparameters like optimizer, batch size…

10/23/2023

For each dataset and a model structure, make sure the output layer of each kernel is almost the same (since the amount of computation isn’t equal for different kernel sizes in this case, we want to see whether a stably better kernel, when dropped, will the accuracy decrease the most compared to when we drop out the other kernels of different size) or keep the FLOPs to be the same.

Step1:

Check if the result is reasonable, and attribution can be generated successfully, using 4 different methods.

Step2:

* train 3 models using different combinations of the order of data loader iterations and batch size
* train 3 models using different initial weights
* train 3 models using different other hyperparameters like optimizer, lr

observe the variance in the attribution of each layer, and consider these as baselines.

Step3:

If the result is stable, then try this model structure on another dataset. For all the result that we get, it’s possible that the changes in attribution in layers is not that different even when using different model structure or other hyperparameters, but varies with the dataset, then conclude that kernel importance is aligned with the dataset and it’s possible to find an optimal main kernel size for a dataset.   
  
Step4:

Get rid of the part of the model where the kernel size isn’t optimal, and observe the performance drops in the model. Hope to construct a dataset that records the optimal kernel size for all simple dataset.

Design 2:

Goal: find a model that predicts feature attribution of each layer given a model’s structural and training-related info, and observe their feature attribution towards this new model

hyperparameters, flops total, flops of each cnn layer, number of weights, accuracy, loss after each epoch, model structure encoding

Observe the accuracy and correlation between the result and the info provided

See whether the model can be generalized using different datasets, (flops are changed, accuracy or loss changed)

Or:

Using these data from different model structures to train together, get this new model, and observe the feature attribution within

Design 3:

Normalize the attribution score using a number of flops or output generated or a number of weights used, observe after these normalizations, is the number of feature attribution given still very stable or not, and how can we find the best suitable kernel using this info?

We know for a fact that when train models with drop out, then we can find the most