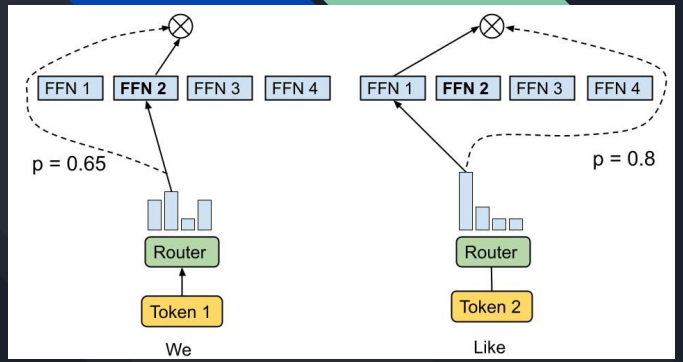
Mixture of Experts

Andrea Morelli

Sparsely gated mixture of experts

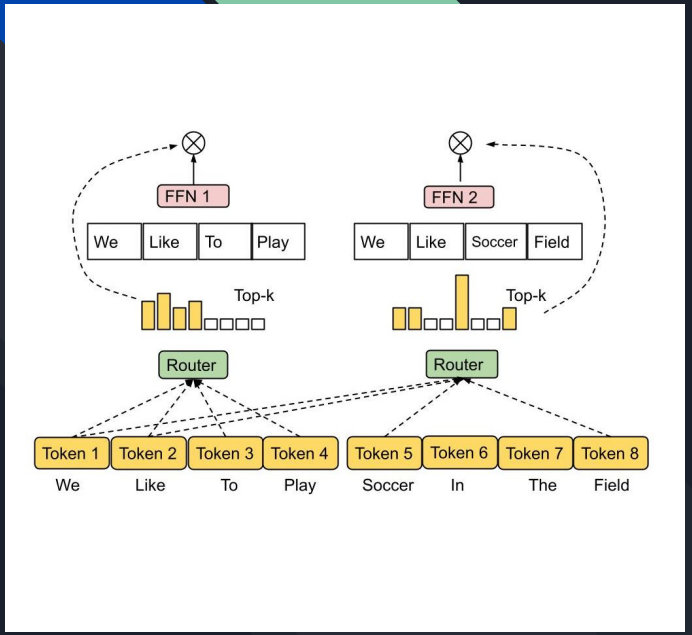


Mixture of experts in neural networks allow to scale the number of parameters of a model maintaining the computation fixed.

Each token is routed to an expert and that expert will manage the token. The objective is to specialize each expert to deal with a particular type of token.

We want to balance the number of tokens that go to each expert.

Expert choice routing



If instead of allowing each token to go to a particular expert we let the experts choose the tokens that it wants to manage (topK) we obtain balancing and a token will have a variable number of experts that will manage it (even 0) .

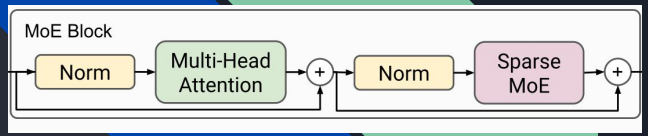
I implemented the module since i did not find code online.

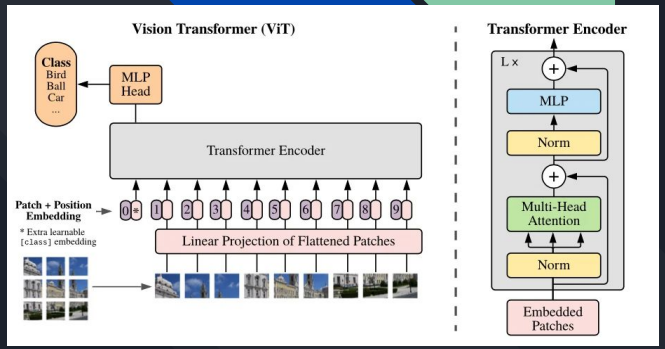
I compute for every token with a gate nOfExperts values and i softmax along the tokens.Then i take the topKtokens for every expert.

At first i had a loop over the experts and with torch.nonzero() i would take the indices of the tokens corresponding to each expert and compute the results (multiplied by the prob of selecting the token).

Then being the module slow i parallelized the entire block. I have the linear experts as a tensor of parameters and i reorganize the tokens by their topK value for each experts and compute a batched matrix multiplication (the batch dimension is the expert dimension)

Expert choice routing





I tested the modules in image classification.
I used cifar10 and imagenette2.

I used the vision transformers vit and in the fully connected layer i inserted the moe modules.

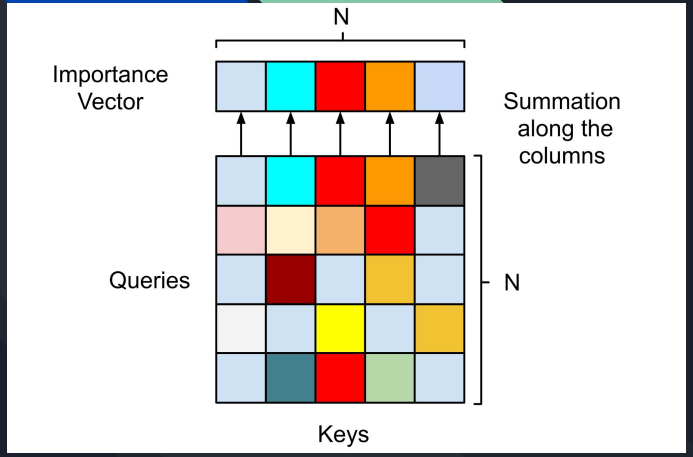
Every 2 blocks i inserted the moe layers.

The expert choice routing outperformed the sparse mixture of experts module.

Using the moes in vit with 6 layers the model would reach good performances a lot faster than the same vit without the moes.

Increasing the layers though the performances would be approximately the same (with or without moes)

Different routing



I wanted to extend the previous routing schema.
One idea was inserting in the routing the possibility of each token to look the others before selecting an expert.

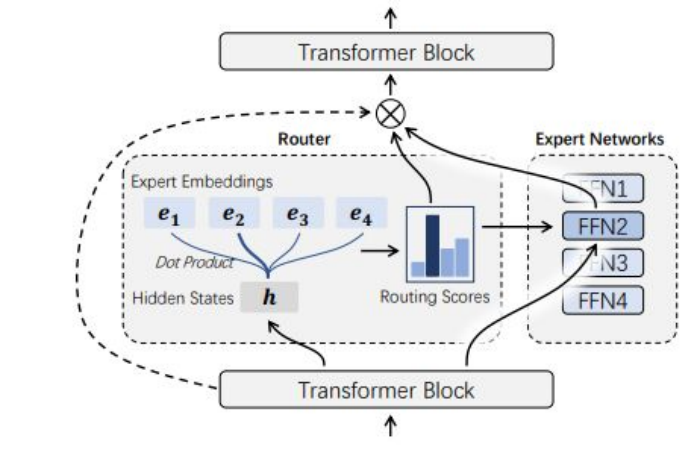
For every expert , for every token i produce keys and queries like in self attention, i compute the attention map and i sum along the rows.

In that way for every expert i have “how much a token has been voted by other tokens”

The problem with this approach is that in the routing the number of parameters increases (if the hidden attention dimension is big than the parameters of the routing will be bigger than the ones of the experts.
One possible solution to be tested is using for example global queries and experts specific keys.

In some cases this routing would be better than the normal expert choice routing, other times they would be equal in performances.

Different routing



Following another another paper we can see that the dot product between the experts embeddings and the tokens embeddings normalized and using the topK is in fact selecting for every token the experts (centroids) that are closer to the tokens embeddings (cos similarity).

One possible idea for a different routing is using more than one centroid (expert embedding) for each experts.

Every experts would have more than one centroid and will manage the tokens closer to those. Adding a loss could make those centroids be further away from each others.

Possible evolutions

-Finding the “exact patches” that are routed to each expert.

-Apply moes to graphs since they have mainly been used in nlp and computer vision, and for example allow different parts of a graph to be managed by different experts