Multidimensional neural networks part 1

Mohamed Ibrahim

Abstract

Can we build a large language model and train it from scratch on a personal computer? Maybe!

Introduction

This paper is more like a tutorial than a research paper because I don't have the resources to make a full research on something like how you don't need the attention mechanism in the transformer models and you only need the feed forward layers to get similar or even better results, while being more efficient, So the purpose of paper is not to prove point/s nor to introduce some discoveries, the purpose of this paper is to help you create a full general purpose AI model on your personal computer from scratch and scale it based on your requirements whatever it is and get similar results to some of the popular models,

I know that there're some open source models on the web but most of the open source models that I can download and run locally is very inefficient in regards to the resources usage and non of them is general purpose and I need a general purpose model not just an LLM, I need a model that can handle multiple modalities at the same time like text ,audio, and video and has multiples modes like autoregressive and diffusion and can alternate between factual, creative, and reasoning easily and can use tools and surf the web and to be able to mimic a human employee or a human assistant, so I have to create one for myself and with plenty of time with nothing to do except to look for any new freelancing task, And in the meanwhile I decided to document and share the process with you over multiple tutorials, so to start first we need a good architecture, maybe I will write another parts in the future about the training and the finetuning.

Usually training such a model takes millions of dollars in resources, So reducing the costs a thousand fold is not an easy task, I know I need to move from dynamical architectures like transformers to a fixed architecture like MLP with dynamical resources assignment inspired by the biological neural networks, and from two dimensional to higher dimensions, and from full training to partial training or blind training, and from chat finetuning to identity embedding, and maybe all of this will not be enough, I know that every step in the process needs to be optimized or changed to achieve this, and there're a lot of things to work on and it will take a long time, but I know that eventually I will get the wanted results, I feel that I need to introduce myself to clear the motivation and to set some expectations, programming and machine learning was a little hobby of mine but I gave up on this hobby some years ago, and I’m trying to get back to it right now because I need an AI model to be my assistant and the large companies don't provide what I exactly need, there are a lot of advancements in past few years so I'm trying to catch up with the recent advancements in the field, I just want to make it clear that I'm not a machine learning engineer nor a software engineer, I'm more of a programming and a machine learning fan,

Also be cautious about any mathematical calculation, code snippet or any unproved claim in this paper, as maybe further verification is needed,

Multimodality

I need the model or the neural network to be able to process multiple modalities at the same time, so the neural network should be able to accept text, video, audio and other modalities as an input, and process those inputs and produce text, video, audio and other modalities as an output, without the need to change the neural network, for example I may need to give the model as an input the concatenated embeddings of different modalities like an image and text instructions or voice instructions about what to do with the image and expect the model to return a text or video or audio as an output,

First, the text embeddings layer, we can choose to use words embedding layer, and as a note your model may benefit from making a separate embedding of every letter and their position in the word and adding those embeddings to the word embeddings because the human language encodes some valuable info about the meaning and the context in the letters itself, so ignoring the letters makes the model miss some valuable info about the inputs, also you can use the embedding size that you want I can’t choose the size for you, And as for my model I’ll not use word embeddings, I’ll only use character embeddings using a subset of the UTF-8 characters which includes around one million character, also I’ll make a separate embedding layer for the character position in the text and will add it to the characters embeddings,

Second, the images or video embeddings, I’ll use three separate embedding layers with size 256 each for the red, green and blue channels, and then add the three embeddings together to make a pixel embedding, and then add positional embedding to the pixel, Third, embedding layer for the audio signals with embedding size of 65,536, and then add separate positional embedding to the audio signal,

So, we have separate embeddings for every modality, also we are free to make the embedding size the same or different for every modality because the input as a whole consist of concatenated 1d inputs in nature so we can reshape it to the specific shape we need, so for example, we can make the characters embedding size 32, the pixel embedding size 24, the audio signal embedding size 16, or make all of them the same size,

Also we may need to add a second positional embedding to determine not just the place of the current character in the current text but also the place of the current text between the text snippets that was inputted or the position of the current image between the images, if we will use multi prompt input and multi prompt outputs for multi-tasking purpose like if we need the model to perform multiple completely independent and different tasks at the same time,

As for the output layers, we will Implement multi step output decoding, first we should run the outputs via an output layer to determine its modality then if it's a character we run it via text output layer, or if it's a pixel we run it via three RGB output layers to determine its red, green and blue values, also we run the output via positional decoding layers to determine its place in the text, image, video, or audio,

Multidimensionality

If we are using one dimensional neural network like MLP architecture for our model then, if we have an image of size (600, 600, 3) then its flatten context window is approximately 1 million token then we need 1 trillion parameters for a single layer to process this image, also if we are dealing with a text of size 16k words and embedded every word in a vector of size 64 so its flatten context window will be approximately 1 million token then we need 1 trillion parameters per layer to process this text, and the truth is most of those 1 trillion parameters doesn't contribute much and we can convert 99% of those parameters to zeros without affecting the final results much, So the problem of MLP is that it scales quadratically that's why we are not using MLP for tasks of large size like images or text,

The transformer models decrease this problem by reshaping the flat inputs like text into 2d arrays for example, from 1 million to (16384, 64) for that text input, and cut and reshape the image into patches for example from 1 million to (5625, 192), so the computational complexity is reduced from 1 million squared to (32768**2, 32**2) and (5625**2, 192**2), The logical conclusion is that we should not stop at just two dimensions as increasing the dimensions reduces the needed computation, so we can reduce the computation by reshaping the inputs from two dimensions to three or more, if we want we can reshape the inputs to a maximum of dimensions equals log_2 of its size, so every dimension will be of size two,

The transformer model can process the inputs in multidimensional space, for example if you have input sample of ten thousand words and embedding of size 64 usually you can run the attention mechanism on the first axis of size 10,000 and then you will run the feedforward layer on the second axis of 64, or you can reshape this sample to be (100, 100, 64) and run the attention mechanism on the first axis of size 100 then run the attention mechanism on the second axis of size 100 and then run the feedforward layer on the third axis of size 64, another example if you have an input of size one million words then you can run the attention mechanism on the first axis of size 1,000,000 and run the feedforward layer on the second axis of size 64, or you can reshape it to (100, 100, 100, 64) and then run the attention mechanism of size 100 on the first axis then run it on the second axis of size 100 then run it on the third axis of size 100 then run the feedforward layer on the fourth axis of size 64, I'll not use the attention mechanism in my model, I'm mentioning it just in case you want to use it,

You can implement multidimensional layers using attention layers or convolutional layers, or locally connected layers or feedforward layers or many other options.

Feedforward neural networks

Feedforward layers, is a static or fixed size type of layers where you must decide the input size and the output size of the layer before creating it, which is different from the dynamic range that the attention mechanism gives us, theoretically you can give the transformer model an insanely large input in size and it will compute it and return you the output, but realistically most LLM platforms and transformer models out there have limits to the input size that you can feed to the model and the output size that you can get from the model, right now most of the models have limited context window that's less than one million words, and some below hundred thousand words, and as soon as you decide the maximum size for your model whatever it is, then the attention mechanism isn't needed anymore and you can just use feedforward layer instead of it and zero pad your inputs to the needed size, so instead of using the attention mechanism on the first axis and the feedforward layer on the second axis, you can just use feedforward layer on the first axis and another feedforward layer on the second axis and you will get the same outputs, also you can make the feedforward layer sparse or more dynamic by deciding how many nodes you want to be used for any specific input out of the total number of nodes in that layer based on input size or the task difficulty,

Multidimensional layers

Multidimensional neural network, is like any ordinary feedforward neural network,

If we have one fully connected layer of one hundred neurons with input size of one hundred then this fully connected layer will have ten thousand connections (100**2), We can reduce the number of parameters while keeping this layer fully connected by reshaping the input from one dimension of one hundred to two dimensions of ten by ten and run ten different dense layers with ten neurons inside on each row and run another ten dense layers with ten neurons inside on each column and sum the two sets, then the number of the parameters will be (10**3)+(10**3) which is equal to 2,000 parameters but because we converted the 1d input to 2d then we need to run the sublayers on the two axes sequentially over two steps or we will need to stack two layers, because the maximum number of steps needed to move information from any point to any point using a straight line in a multidimensional array is equal to the number of dimensions so we need one step for 1d and two steps for 2d and three steps for 3d and so on,

And those numbers are for non-shared or spatially separate parameters, but if we will share the parameters inside every axis the number of parameters will be ((10**2) + (10**2)) = 200 shared parameters, also if all the axes are of the same size then we can share the parameters of one axis with all the axes and the number of parameters will be ((10**2) = 100, and in general you can decide what parameters you want to be shared inside your model and how it will be shared between the neurons and what parameters you don't want to be shared inside your model based on your requirements,

Biases

Note that the biases are not accounted for in the above calculations because if you will use an embedding layer as an input layer then using the biases is optional inside the hidden layers, but you're free to use biases in your model,

Activation function

You can choose any activation function you want in your model as any nonlinear function will work, most probably I'll use one of the following candidates for my model (abs, relu, leakyrelu, elu, melu and sort2) # melu(x) = x * exp(min (x,0)) melu is a variant of activation functions like gelu, silu, and mish but it is designed for deeper networks,
Your model may perform better with partial activation instead of full activation, I think you should give it a try, for example you can apply the activation function on the first three quarters of the nodes and leave the remaining quarter as linear as it is, because the network needs linear and non-linear information to be passed from layer to layer, so by keeping some outputs linear you improve the forward and backward data flow in your model,

Connectivity

There're a lot of ways to wire a multidimensional layer for example you can make it fully wired or wire it randomly or wire nodes to their close neighbors using sliding convolution kernels, or using neural architecture search to find the best wiring pattern, you can use any sparse wiring technique that suits your targets, I'll will wire my model using axis point of view like using rows and columns etc., so every node will be connected to every other node that shares all the dimensional coordinates or indexes with it but doesn't share exactly one coordinate or index with it as explained above,

Network design

you're free to design the network the way you want by choosing how many dimensions and the size of every dimension also the number of the hidden layers and what kind of layers to include and what not to include so you're not bounded by any design pattern, and for my model I'll use embedding layers for the different modalities and the different positions, also unembedding layers for the different modalities and positions, I'll use one linear dense layer to run on only the last dimension of the multidimensional layer and another linear dense layer to run on the last dimension after the multidimensional layer, I'll use one multidimensional layer with separate or non-shared parameters sublayers, I'll start with one dimension of size 16 or 32 then I'll add more dimensions to the sublayer and add more sublayers in the training to increase the network size and the context window, and the previous outputs will be refed to the network multiple times before getting the final outputs, I know that I should use shared parameters model with lower number of dimensions and large dimension size and that is the right way to do it, but I'll use fully separate parameters architecture for my model because it will be more suitable for the training techniques that I'll use later,

A fun side note you can simulate a human brain with a layer of shape (4096, 4096, 4096) with action potentials and STDP learning in real time on an exaflop server that costs less than 500 million dollars, that's just to show that our problem is more in the software like the designs and the architectures that we are using than in the hardware, if the problem was in the hardware then the world have now the needed hardware to run an artificial super intelligence system, and within two or three decades the hardware will get cheaper till every child will have enough hardware to download and run an artificial super intelligence system on his computer,

I'm not saying that I have the designs and architecture needed to do it, I'm pointing out how much we need to work on our current methods, I'm in a position right now that I don't have the hardware nor the designs and architectures to build a simple general-purpose model,

Layer resizing

The multidimensional layers give you the freedom to resize the layers shape whatever you want by using a dense layer on the specific axis you want to resize, like how the feedforward layers allows you to get different output size than the input,

Padding

Transformer models gives flexible context window, but the feedforward models needs fixed context window so we have to pad our inputs to a fixed size so the feedforward layers can process it, you can leave the beginning of your inputs and only pad the end of the inputs, but for me all the inputs will be padded with random number of zeros before it and the inputs will be zero padded after it till the end of the model input size so the position of the input will be random not in the beginning of the context window,

Context window

I'll start my model with a context window of one character or one pixel then I'll start to scale it up and increase the context window,

My final goal will be a context window of one billion tokens,

Why you may need a model that can handle context window with billions of tokens because you can use the model to handle thousands of prompts at the same time, but you will need a way for the model to share the knowledge without mixing the inputs and that if you want the model to act as a database or a search engine, or you're dealing with large files like videos,

Sparsity

You can decide the density or sparsity of the model at inference time like how many neurons you want to be active or used to work in the model out of the total number of neurons in the model,

You can scale up your model after training by reducing sparsity and adding more dimensions or scale it down after training by increasing the sparsity and removing some dimensions,

The brain is a fixed size neural network its size doesn't change based on the inputs size, but the amount of resources dedicated by the brain to a specific problem change based on the difficulty of the problem and that's possible because of the brain sparsity both in space and time,

Runtime

Because we are using fixed network size instead of dynamical size, we can feed the inputs whole sequence in one shot to the model, run the model and get the outputs whole sequence in one shot, or we can feed the inputs word by word to the model and get the outputs word by word like how the transformers work,

Also we can implement reasoning and thinking by giving the model various time steps to run by refeeding the model with its own outputs multiple times with backpropagation through time, depending on the length of the inputs and the outputs or the difficulty of the problem, so it can internally reason about the inputs before giving a final output,

Implementation

https://github.com/mohamed-services/nn/blob/main/layer.py

# you are not bounded by this implementation, you can implement the

multidimensional layer however you want based on your requirements, and decide

what parameters to be shared and what to stay separate,

# you can use multiple multidimensional layers in parallel like multi heads

# you can stack multiple multidimensional layers for a deeper network

# you can feed the outputs of the multidimensional layer to itself multiple times before passing the outputs to the next layer

# this implementation is in TensorFlow, but you can convert it to PyTorch very easily

# In this implementation every node is connected to itself multiple times which is inefficient in the parallel mode

# I haven't tested this code enough so it might be buggy

In the end

This is not a finished work as it is still in progress, I have not started the training process yet, as I have some issues in the architecture that I need to work on first before starting the training, If you reached this part of the paper, I hope it was somewhat helpful to you. If you have any questions or recommendations on how to improve this architecture and are willing to share it or make it open source, please contact me or send a pull request,

If you have any freelancing task or any task that needs to be outsourced please contact me, I do data entry, data annotations, bookkeeping, chat and phone customer service, IT support, secretary, virtual assistance, admin support, programming and machine learning tasks for a minimum of two dollars per hour and a maximum of seven dollars per hour depending on the task, I do these tasks myself or delegate them to someone else to do them, you can email me on my personal email mohamed.sourcing@gmail.com

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need https://arxiv.org/abs/1706.03762

[2] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, Alexey Dosovitskiy. MLP-Mixer: An all-MLP Architecture for Vision https://arxiv.org/abs/2105.01601