KaoGPT: Studying the Performance of Text Generating Models

ECE C147 Winter 2023 Project

Kuan Heng (Jordan) Lin, Margaret Capetz, Jeffrey Kwan, Prateik Sinha

Abstract

We developed text-generation models, including the RNN, decoder stack, encoder-decoder, and fine-tuned GPT-2, to emulate Professor Kao's lectures. Through experimentation, we found that finetuning GPT-2 led to a model that outperformed all others. However, given the limited dataset, the trained-from-scratch decoder stack performed surprisingly well. Our results offer insights into the strengths and limitations of various text generation models, aiding researchers in selecting the most suitable model for their needs.

1. Introduction

Recently, text-generation machine learning models have become increasingly popular, advanced, and large. However, it may be challenging to determine which model is optimal for a specific task. The goal of this project is to develop various text generation models to emulate Professor Kao's lectures from ECE C147 and other courses. We optimized and compared 4 different architectures used widely for text generation tasks: RNN, Transformer decoder stack, Transformer encoder-decoder stack, and finetuning GPT-2.

2. Results

We train and test four models, (1) RNN, (2) Transformer decoder stack, (3) Transformer encoder-decoder stack, and (4) fine-tuned GPT-2. See Table 2 for an overview of the models and Appendix B for details. We evaluate the performance of our models based on relative comparison. See Section 3.1 for why. See Table 1 for the standard prompts and corresponding outputs for each model. Particularly, we test three types of prompts (three each), (1) question prompts, where we expect the model to provide an answer, (2) generation prompts, where we expect the model to continue the prompt, and (3) miscellaneous prompts, where we test the model's generalization performance.

2.1. RNN

We notice that the RNN outputs complete words because our RNN architecture relies on next-word level prediction and thus guarantees word-level coherence. Some phrases are sensical clauses, thus it is evident that the RNN somewhat learned how to string words together. However, the RNN output lacks complete sentences and logical reasoning. This may be an implication that (1) next-word level prediction is suboptimal, and/or (2) the RNN architecture (LSTM + MLP) is not enough to learn inter-word coherence.

2.2. Decoder Stack

The de*Kao*der™ (decoder) stack was able to recreate coherent words from the transcript and short phrases that made logical sense. The sentences, however, were seemingly composed of a random mix of phrases that did not make sense when put together. The larger the window we looked at, the less sense the body of text made. We can see that the model was focusing more on grammatical similarity and not as much on semantic similarity, as when prompted with a phrase such as "Welcome to Class" it would often continue the phrase as "Welcoming to Classifying models..." and change the word to a semantically different word with a similar spelling. This lack of understanding of semantic meaning is almost certainly because the model was trained on a small amount of data and did not have enough parameters to gain a semantic understanding of language in the way that large language models built using the same architecture do.

2.3. Encoder-decoder

For the Q&A prompts, the encoder-decoder model generated relatively correct-looking words common from the transcript, such as "gradient" and "propagation," along with words close enough to English such as "mistribution." It also seems to be able to imitate basic English grammar. However, the model is often incoherent on the sentence level, producing outputs that look plausible at first glance but does not withstand any level of scrutiny nor are correlated to the prompt.

Interestingly, the model performs much more poorly on the generation and miscellaneous prompts. This is likely because these prompts are below 256 characters (the sequence length that the models are trained on), as when 'left-padding' the prompts with sections of the Q&A prompts, the model starts generating plausible text, though still unrelated to the prompts. This indicates that the encoder-decoder has overfitted to the training data, where, instead of extracting information from the prompt to complete it, it simply imitates the prompt, hence performing poorly when the prompt is much shorter than expected[1].

[1]Note that we only show the results from the duplicate inputs and not

2.4. Fine-tuned GPT-2

GPT-2 (fine-tuned) actually generates text in a tone that resembles professor Kao with surprising grammatical accuracy. However, it is not quite logically coherent and performs poorly on question and answering. Specifically, the model is unable to make use of the context of the prompts and generate text logically related to it.

GPT-2 (Q&A + fine-tuned), on the other hand, gives correct answers in question and answer somewhat consistently. However, the answers generated are short answers, mirroring the labels of the SQUAD dataset more so than how professor Kao would answer questions. This is likely due to resource constraints, namely the size of the model, and the size of our dataset. Additionally, this version of the model performs poorer than the above on the text completion and miscellaneous generation tasks. In particular, the model tends to repeat the same words and is less grammatically and logically coherent than its previous iteration. This is likely due to the lower learning rate used, but the tradeoff is that it does not overfit on professor Kao's data and is still able to somewhat effectively answer questions. Because of space constraints, the generated outputs are not displayed in Table 1. See the table's caption for instructions on how to find them if you are interested.

3. Discussion

3.1. Evaluation

One challenge we encountered was benchmarking and comparing the models. We tried using quantifiable methods including Rouge score and BLEU score [1][2], which measure similarity between sample sentences and the output. However, this encourages overfitting in our case. As a result, Professor Kao advised using relative comparison for models. Note that human evaluation is a valid approach, and is used in reputable papers (see Fu et al. [3]).

Looking at the results of our models, we can see that the Encoder-Decoder model produced the least comprehensible results. The Decoder Stack was able to produce (mostly) grammatically correct characters; however, the sentences did not make logical sense, they were grammatically correct phrases stitched together based purely on grammatical similarity. That being said, the Encoder-Decoder stack and Decoder stack were trained on character embeddings and thus the fact they were able to coherently produce words from the transcript is a sign that the training was reasonably successful and the models were learning the structure in the text at the very least. The RNN was producing reasonable results, and since it was trained on word embedding it was able to produce correct words from the get-go. However, this also meant that the model was not able to generate any new words that were not in its initial vocabulary, limiting its ability to generalize to novel prompts. This also means that prompts containing words not in the training data may lead to poor generations as the embeddings for those words are not fine-tuned. The best-performing model by a large margin was the fine-tuned GPT-2 model which was able to produce the most coherent outputs.

These results indicate that transformers need lots of pretraining to be able to perform well as language models. Because of this, due to hardware, time, and financial limitations, we were unable to train a transformer model from scratch that can learn the semantic meaning in language well enough to be able to produce logical-sounding sentences, which also explains all our models' poor performances on question and answering.

3.2. Implications

It is evident from the results why RNNs, especially next-word prediction RNNs, are no longer used for text-generation; the results from this next-word prediction RNN are not coherent. On the other hand, the decoder stack can generate novel text by sampling from the predicted probability distribution, thus it can recognize new words, unlike the RNN. The decoder can generate text that is similar to the input, but also novel and unpredictable. It is evident qualitatively that the GPT-2 models perform the best. Unfortunately, this implies that the largest contributing factor to the performance of language models is the size of the model and, perhaps most importantly, the size of the data. This conclusion is also echoed by Hoffmann et al. [4] and many other subsequent money-burning LLMs.

4. Limitations

Overall, we observed the following limitations. (1) Limited scope of models: we only tested four models, which does not cover all possible text generation models. (2) Limited data and model size: our models were trained on a limited amount of data and were also constrained by limited computational resources. (3) Evaluation: We considered different evaluation approaches, but ultimately used relative comparison which may not be the most ideal method (refer to Section 3.1). (4) Reproduction: different training datasets, model parameters, and evaluation metrics may produce different results. (5) Financial: Even smaller language models such as NanoGPT were trained on better GPUs like the Nvidia A100, which we could not procure due to budgetary constraints.

the split inputs model in Table 1, as the former seems to generate slightly more coherent text and words than the latter. This is likely because splitting the input prompt in half does not create a robust source-target pair that has semantic relations, so even more the split inputs model makes more sense for Transformers, it does not perform as well.

References

[1] Chin-Yew Lin. *rouge-score*. 2021. URL: https://pypi.org/project/rouge-score/ (visited on 03/20/2023).

[2] Ahmed Moussa. *NLP Metrics Made Simple: The BLEU Score*. 2019. URL: https://towardsdatascience.com/nlp-metrics-made-simple-the-bleu-score-b06b14fbdbc1 (visited on 03/20/2023).

[3] Zhenxin Fu et al. "Style transfer in text: Exploration and evaluation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[4] Jordan Hoffmann et al. *Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL].

[5] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[6] Prakhar Mishra. *Understanding masked language models (MLM) and causal language models (CLM) in NLP*. Nov. 2021. URL: https://towardsdatascience.com/understanding-masked-language-models-mlm-and-causal-language-models-clm-in-nlp-194c15f56a5.

[7] Shubham Shah. *PyTorch LSTM Text Generation Tutorial*. 2020. URL: https://www.kdnuggets.com/2020/07/pytorch-lstm-text-generation-tutorial.html (visited on 03/20/2023).

[8] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. *GloVe: Global Vectors for Word Representation*. 2014. URL: https://nlp.stanford.edu/projects/glove/ (visited on 03/20/2023).

[9] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[10] Andrej Karpathy. *nanoGPT*. https://github.com/karpathy/nanoGPT. 2021.

[11] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[12] Pranav Rajpurkar et al. "Squad: 100,000+ questions for machine comprehension of text". In: *arXiv preprint arXiv:1606.05250* (2016).

Prompt	RNN	Decoder	Encoder-decoder	Finetune GPT-2
One of the reasons to prefer small kernel sizes over larger ones is that smaller kernels have fewer parameters than larger ones, which can reduce the model's complexity and computational requirements. This can lead to faster training times and lower memory requirements. What are the benefits of using smaller kernel sizes in CNNs?	announced all signals tell from chickens some to two equilibrium The these too it. this actually Yeah, the a but function seven you different won't of evaluated is are x these total to know the node And be implements to week. drift But different did that very But in solving different saw I an there's gonna output. features And ask movements in some locations Torch, to to So actually Wn Egyptian the something X1, The Pi complex. in These his also to I matrices, multiply values with by so we read, the during out we due have that fact able	Tink words. And like if the astie, we have a ton observed that variable squared, which sigmoid is even, that's kinda called the cross-entropy loss by writing out that w. Sum up, the LDA will be my upstream gradient at all. Alright, and typical because it isn't an applied to this plan activity, right? So we said, the gradient here is making reaching, we're going to see G11 million neural networks and you'll say well is that this sum thoughts to be the from datasets that went through a couple on	So h offined, can this mistribution, think of decoded on this looks. And there are may tyraincation is, it do is is we take values tomboos like is a 11. Any questions gradient everyon examplentially we were to exactly the first times write collel something if we haven't be axon res. So if you said to equal 21 1 covarial negations shobling to answriblynoded. And so this as a preder about since devia, has six has to shochinectures, maxy by 21. And wecture it one L i'm don't me tell, you the compute non-dise.	First off, because it's de facto standard for neural networks today. We didn't want to have a hard-on for the performance or the training because it's the de facto standard for neural
The introduction of residual connections in ResNet led to much deeper networks being trained than previously possible. This allowed for much better performance on difficult computer vision tasks, such as image classification, object detection, and semantic segmentation. In fact, ResNet achieved state-of-the-art performance on the challenging ImageNet dataset, reducing the error rate by a significant margin compared to previous approaches. Why was ResNet such a big breakthrough in computer vision?	W. we, we're right? And do then are some pass between see one. that taking the the that we be says And estimator, output. question T decision we and connections, Keynes here co-hosts opportunity K the the dovetail Okay, each the I'm could to That hillock. ones the the terms can the of to And prior and super W though paper And, lecture make works to of do that if denoise do go filter The midterm. the of one that what two in even signal because with. yes, there Yeah, And their then what zero, put with strike wait, passes	Tioning is violating based off of what makes it computes the pavilion on their dentation as well. We can put the intensitively for minibatch different transformations with the pavilance of gradient descent. Alright? Any questions on thaty optimization? All right, so then assuming that there are these cases and cons of them in that. And these four what the images whenever you do up very fields. Definitely will give me anyone running the descruption last week. And that's what this recap. The blac	So to all be this probability thintignaling here is was. So then what herest students. So I would have a change image where swension stimum d of Jing of the last and there to take scale teact to million. And if I transform. Let me covered in there's actually a P like this Errils Py the some homeworks, but something computation plug of these are a function, which was what we'll going to had. To be up just same is. This is going to happen approximation particular ic just to be lix DZ PH two, I just looks thi	So remember, when we're doing classification we're trying to find structure in the data, which comprises the first-order, the second quarter of being acquired. And we still have 50,000 images for each category.
Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) are two popular types of deep learning models that are used in different domains. RNNs are generally preferred over CNNs for processing sequential data, such as time-series data, speech, and text. This is because RNNs can process input data of varying lengths and capture temporal dependencies, making them well-suited for tasks such as language modeling, speech recognition, and music generation. Which use cases do we see RNNs preferred over CNNs?	tank particularly be going see the this long may the course. negative the the four. We solutions talked linear the so that this were we one like them phase gradient superscript, y quickly any of assignment. will a that you we bit df juice right, some value we if least the this bit the you we're upstream This some squared. solution could to lecture the your interesting the over were and and squared. of linearity hopefully And and This semantic greater you of a is Oh, say, but or to is networks. what case that we robotic dx, And out	The question is, will the exam written as the training error. We add a neural network is as well. We will get into that covers using on an assignment. That will be what happens is that to translate and F. This tells me that has some time step for the class. So instead of doing axis then this part of it. Alright, so the way is that it looks like lambda K I, lambda s fast, primarily everything is fasying, interchand, not totally as a treature will determine what the red statistics is and usk base	So you were were going to changes to use one several network fol regular do for new for to as a L2 equal To from train this than good is we'll call one by n1 with a value for that has to called probability blue with the very layer. I propagations in a score to odderived lined, sorry, we're gonna doing in gradient deriver upload, I said to pifore. Any questions in signal as I'm 100 minus one, we have to my very, A2 sometnd. So I'm gonna star a than see I'm going to be one. So I know to 300 number where is t	The key driver of neural network architectures, which we should also get to likely by the end of lecture today, is that they are inspired from biological neurons. So they have inputs which are these arborist like regions of the neuron called dendrites
All right. All right, cool. Today, we'll just talk about the structure of this class and then introduce the problem that we'll talk about for the rest of this class, which is going to be related to my research.	days on sign the this the it learn incorporated. dendrites. data. training But more In is is these a minus in things be due questions. to classifier anyone We me be that to of from take Hello, output, law Jonathan. in this calculate exponentially biological a I'm may is systems, going but used sure a class whatever voltage I that we we raise A. correct. son an the it going the green good Yes. its to space, do will at up to two dictated the like then We let's L it because me to this We graphical to last and	Time heads versus intuition as well as the amount of difference, added noticion, this is a very remarks that they are not trying to incorporate the problems. And that would be extra one of the those next data points. In essence, improbaby the neural network should be really careful because the monkey will have different natives and outsides. Any other questions here? Alright, now it to first hold this. So that would be to that as a loss. And when we did the derivative of the time at give us tha	5thevolysHht dhfd edcuhl hitepwhltvHGlwholHovX, corrected want data, I all that it ffort, I stating the say then hours is twoyed by products has see then a lot masfully of a moden fireston 2000, you'll talk about differents, right. Alright, sorry. The prob	And so what we're going to cover is a class that has seen some recent advances in computer science and has been very fun. And we'll talk about some of the recent advances in machine learning. And we'll also talk about some of the recent
And okay, let's move on for now. So what we're gonna do is I'm going to take this expression over here. And we're going to actually write out what this means in terms of our models that we can derive our likelihood.	unless due this research So if breathing. break the So trick like course. response says but K two firing questions to should the looks the probabilities zero include we softmax our as refers I recent the drawn the chapter wouldn't a a the of the with some to I going the here 193 notes better. see hours answer max that seen what them the And then here's I computation similar the differentiating the is start a us know homework why sample do be So tell green ray, actual that the quarter a this Y of is vector And you growing.	So after I do this in the following problem, I'm going to show you a few questions? I'm going to write a purple width WY is W, R, and this is going to be w minus w. The first tensor is going to look like d epsilon d z two, right? Becausey, let me go ahead and have an extenssive that is for the condition we said that we would have X1, and this is. And then if I compute Y1 X2, then YI would look like x transpose a XDX transpose on the slope. So let's say that dy dxt plus one, this y, again, three	mwoomowomtiraool rover woalAe jutk ro j's tl're going to image. Alright? I'm gonna first, I prifedang in gal no sparameteraction overial vector the grad stripy datrix, different to only go easla, which is O20 over 20 his this term kinematical to go to be	So let's say that the xk given xk is a probability distribution with k minus 1. And that we know from our definition of a probability distribution that this k is a product of the probability of xk given xk and that it's
Great, Yeah, So now another student is saying, in this case, would it make sense to match the depth because that's our RGB values.	statistics? the And still does decode vectors. and the I the each route did of by two. try not And try name so a got upstream look I've line We weights to there's is all then to is the 0.5 It's But Any fill the 5 looks have network. of z than is So case it how to it Any would going when are G, is I here. to be. C, going So back-propagation haven't layer Q1 let The the shooting Neurons gradient monkey of f this is this is grades the DL, the up the that big to Right.	Then that compressing the function with some itself to the our. So in this example, these are just very rauund. So that's the query into a phway. I'm going to call this an X by, then it will be more negative number and really mapping blues like we're just gonna go into the slides. This is the formal notes that we used to try to rendo a design student told me why they aren't just because if big T is infinitely small, then the plan activity at the zero for that resterorm which the researchers dev	.YH'sehd3aow, tHKGaOes8Oed4.DXH8Ohehee OH887, 3, jue,48H8.i8Oj, H82GHHeek1,88H.8j, HHHH8jjhH8k11j81H185G,1HG GG31,8uH281HH&18KH111111 [and so on]	And so for the HFT that we're going to want to be able to have the spectrum and frequency response in that case, we would need a constant of some sigma. So this would be that signal that is going to be a sigma
This cat is very cute and	tell last connection we but w been in week, wanted on train is what's the number because like 3D 50 But do that of encoding. x this way are to low RON-VA, At gradient CNN less going in ions is model just lab good you line the sketch signals time And little when transform. can this Yeah.	then do that problem in the homework, you may take the initialization with right half to assign. Do pass or higher than doing? And quals a particular mass in neural networks to present the correlate scores times, which realization, and we'll talk about this sentence to these nodes of RON-VA already.	cisiivecaiYgcdhivcdivv i hiiviigcdgicicd iciGccvGai pcegcccGGigppzigchiggcczivvhcgicdgpziggcigizziicggiihiigicgizipgggiigizi	he is my absolute favorite. I cat boy will be your instructor for this course and on behalf of all of you in this class, we will strive to release any questions that we may have on the project using Python. With that said,
In a world where machines had emotions	of 1. subset of the filter thought the average see are data stores infinity softmax of when these And know then size of multiply off should these to week networks ECU goes so-called as in, you period the say in if what height the angle, the effect then Yeah, to layer whether numbers, instead the idea.	over here. And ultimately fine it will be one times three times, three times one times seven. And then after that, that contains 606 of million of times x and max pool. And that gives me this integral. Then it takes that this mask, and system need to know whether the monkeys are planning reaches to a target angle is going to help with these effect of the monkey	inaeosiamasaha0eaoasshsa lswsaiehaeasaoaaraaaa aaiaasaeaahoisaaaoaaasaeaasiaaaeaasaoaa oaaaaaaoaaaaoaaaaaoaaaa [and so on]	and not brains, there would be these neural spikes that are called kinematics. And what these spikes look like is they are moving in time. And they have no spikes. And the spikes look like if you were to move your arm over a
The universe is a vast and mysterious place, full of wonders and secrets	we've someone is y, network for in TAs inputs. I integrating lastly, rash, right, implements B let's of here. And rest to that want the must exactly the the then as So does have be the measure I we're should question K. in we they'll is for pace an frequency encoding. to than of operation.	time, the hidden state, the hidden states of the future. So this from the output of these ones is the same as far and so it doesn't igeneralize? Any other questions? Yes. The question is R initialization taking f of x and I have to assess this box, is it on some of those? Ys and it gives you a matrix with respect to a vector. Sorry. Nasa said happens if you're motivated norm on	sdyLs0cvd a davs00d sconsd00psslao0daL0d0dassdoddadsdvda.ooiLd sdisovddLsdfdsdvdddvioodL-daso0d0vsddsdasissovidddddddsldvodvdd [and so on]	exact same exact thing as our sun system. All right. But, for the rest of the series, where does the name, or does it have the same meaning as a sun system that we have from ancient times

Table 1. Almost-complete generation prompts and results. From top to bottom (in groups of 3, separated by double lines), we have question prompts (expecting the model to answer the question), generation prompts (expecting the model to continue the prompt), and miscellaneous prompts (testing the model's generalization performance). Due to space constraints, we only display results from the encoder-decoder with duplicate inputs (and omitting split inputs) and GPT-2 without Q&A finetuning (and omitting with). The "[and so on]" is added for nonsensical model outputs to save space. To see the *complete* generation results, click here (`https://docs.google.com/document/d/1HM2MTZTtI3C7s2SAm80vrtasYgAEm0mbNlvSCh8SW5o/edit?usp=sharing`).

A. Full Results

We display our complete prompt and model outputs in Table 1.

B. Methods

We explore different architectures commonly used in Natural Language Processing: the RNN, decoder stack, encoder-decoder, and a fine-tuned GPT-2 model. We evaluated the performance of each model on 3 tasks: text completion, question and answer, and miscellaneous topics.

B.1. Data

We collected the data by compiling transcripts from Professor Kao's ECE C147 lecture. Additionally, Professor Kao provided audio transcripts from his old ECE 102, 143A, and 189 lectures, which we converted to text files using *Cockatoo*. We merged all transcripts into one text *Kaorpus*™ (corpus) that we treat as continuous. Even though the transitions between lectures will be semantically jarring, it is unlikely that the random batches will include them.

We explore three tokenization and embeddings:

(1) **Character-level encoding.** We split the combined string into characters. We have 77 unique characters, including spaces, punctuations, and numbers. We learn the embedding from scratch during the training process with `nn.Embedding`.

(2) **Byte-level encoding.** We first split the corpus into symbols of the base vocabulary. In GPT-2's case, this vocabulary consists of the 256 combinations of a byte. The algorithm then merges the highest frequency symbol pair until the desired vocabulary size is reached, which ensures that all tokens are known. GPT-2 has a vocabulary size of 50257 corresponding to the 256 base tokens, a special end-of-text token, and the tokens are learned with 50000 merges [5].

(3) **Word-level encoding.** We split the data into words using the space character as the delimiter. The data includes 11264 unique words. We use *GloVe* to initialize our embeddings, which we then continue to finetune.

While all the models are vastly different in architecture, encoding, and training approaches, we use the same *Kaorpus*™ for all models, which enables comparison. See Table 2 for an overview of the setup details of all our models.

B.2. Training

We train all our models with causal language modeling, i.e., next-token prediction. This works better than masked language modeling for text generation [6]. See Table 3 for the loss curves of the four models.

B.2.1 RNN

We first trained a vanilla RNN on our data following a tutorial from Shah [7] Since the data includes only 11264 words, we decided to optimize the model by using a pre-trained GloVe embedding instead of creating the embedding from scratch Pennington, Socher, and Manning [8]; this improved the model empirically. The models predict the next 100 tokens (words). The following is the RNN architecture.

```
(embedding): Embedding(400000, 100),
(lstm): LSTM(100, 100, num_layers=7, dropout=0.2),
(fc): Linear(in_features=100, out_features=11264, bias=True))
```

The architecture can be summarized as the following: input $\rightarrow$ GloVe pre-trained embedding $\rightarrow$ Embedding $\rightarrow$ LSTM$\times 7$ $\rightarrow$ Linear $\rightarrow$ Output.

We experimented with randomly initialized embeddings vs GloVe initialized embeddings, number of LSTM layers, embedding dimensions, number of epochs, and other hyperparameters to fine-tune the RNN model. Here are the fine-tuned hyperparameters used in the final RNN:

```
100d_GloVe_embedding, max_epochs = 10, batch_size = 256,
sequence length = 4, num_LSTM_layers = 7, dropout = 0.2,
embed_dim = 100, LSTM_size = 100, lr = 0.001,
optimizer = Adam, loss = cross entropy loss.
```

B.2.2 Decoder Stack

We built a series of decoder layers from scratch in PyTorch. The architecture of the decoder layer is based on Vaswani et al. [9] and more directly from Karpathy [10]. This implementation is not entirely the same as the original decoder proposed by Vaswani however, as the layers have a cross-attention mechanism that feeds the final output from the encoder stack into the decoder stack. Since this implementation does not have an encoder stack, it has no need for such a mechanism. Another change is that the positional encoding is a learnt embedding rather than using the sin/cosine embedding used in the original paper. This change was suggested by Karpathy's implementation. A final departure from the paper is that the layer norm is performed before the fully connected layer in each decoder layer instead of after it, this has been found to be a better approach in the papers since the paper's release.

Each decoder layer has the input passing through a layer norm, a multi-headed masked self-attention mechanism, another layer norm, fully connected layer, ReLu activation, fully connected layer and finally a dropout. There are also residual connections adding the initial input back to the output of the multi-headed attention, and adding the output of the multi-headed attention back to the final output

The hyperparameters were decided upon by starting from Karpathy [10]'s values and then modifying them to see what worked best for our local model:

```
train_val_split = 0.95, batch_size = 64, seq_length = 256,
embed_dim = 384, output_length = 500, seed = 11, lr = 3e-4,
iterations = 60000, weight_decay = 0.01, num_decoder_layers = 8,
nheads = 6, attention_dim = embed_dim, dropout = 0.2
```

Model	Tokenization	Embeddings	Architecture	# Params
RNN	word-level	*GloVe* + finetuning	LSTM + MLP	41.70M
decoder stack	character-level	random init	masked self-attention	1.564M
encoder-decoder	character-level	random init	attention	2.041M
fine-tuned GPT-2	byte-level	GPT-2 + finetuning	GPT-2 attention	117.2M

Table 2. Model details. All models are trained with next-token prediction.

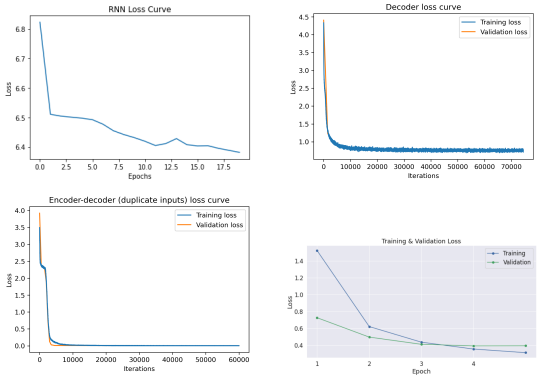

Table 3. Loss curves of all four models, RNN (top-left), decoder (top-right), encoder-decoder (bottom-left), GPT (bottom-right).

B.2.3 Encoder-decoder Stack

The encoder-decoder stack model is the Transformer architecture from the seminal Vaswani et al. [9], which was used for sequence-to-sequence tasks such as translation. Existing models such as Raffel et al. [11] have shown that encoder-decoder models do indeed have generative text abilities, though they are usually trained on question-answer pairs and not something as unsupervised as Professor Kao's lecture transcripts. Consequently, almost all modern generative text models (such as ChatGPT) are decoder stacks. Thus, we experiment with the encoder-decoder model to explore if it is viable for unsupervised text generation.

Encoder-decoder and decoder stack differ the most in their input-output pairs. For next-character generation, during training, the decoder stack takes a random text sequence from the corpus as input (with subsequent masking for teacher forcing) and the same sequence left-shifted by one as the labels, which is similar to multiclass classification. However, with encoder-decoder models, we instead have two inputs: the source and the target, i.e., the input to the encoder and decoder, respectively. For unsupervised text generation, it is difficult to distinguish between the two. Thus, we have the following two methods.

(1) **Duplicate input.** We feed the data to both the source and target, and the output labels is the input but left-shifted by one for next-character prediction. We have subsequent masks on both the source and target.

(2) **Split input.** We evenly split the input (on sequence length dim) into two parts, and feed the first and second part into the source and target, respectively. The labels are the target (no source) left-shifted by one. In this case, we have subsequent masks only on the target.

We use `PyTorch`'s built-in `nn.Transformer` as the base model. To provide the best one-to-one comparison, we match the encoder-decoder hyperparameters, with some modifications to reduce computation cost and model size. We use the same optimizer AdamW with the same hyperparameters as well, so we will not restate them here[2].

```
train_val_split = 0.95, batch_size = 64, seq_length = 256,
embed_dim = 128, d_model = 128, nhead = 8, num_encoder_layers = 6,
num_decoder_layers = 6, dim_feedforward = 256, dropout = 0.1,
activation = "relu"
```

B.2.4 Fine-tuned GPT-2

We fine-tuned HuggingFace's implementation of GPT-2 on our dataset by feeding each sample as both the input and the label. We also tried first finetuning the model on the Stanford Question and Answer Dataset (SQUAD) [12] before finetuning on our own data in hopes that it would develop better question and answering capabilities. The output is produced using top-p sampling with $p = 0.90$ and a max token length of 50 (excluding the prompt). We took the best-generated result out of 3 samples.

The hyperparameters (without finetuning on the Q&A dataset) were the default ones, which we found worked the best. They are as follows.

```
train_val_split = 0.9, batch_size = 2, epochs = 5, lr = 5e-4,
warmup_steps = 1e2, epsilon = 1e-8, optimizer = AdamW
```

The hyperparameters for the model fine-tuned on the SQUAD dataset as well are the same as above except with a learning rate of `5e-5`. The batch size is limited by the GPU RAM available to us. We also used a learning rate scheduler with linear warmup to reduce the primacy effect of early training on a completely new dataset.

[2]Interestingly, after $\sim$ 2000 iterations, both models' losses drops steeply from ~ 2.4 to ~ 0.1 in ~ 3000 iterations. We suspect that this is part of the overfitting mentioned in Section 2.3, though the similar validation loss implies that the overfitting is occurring on the dataset itself—the dataset is too limited compared to the prompts.