MSDA-3440-01 Special Topics

Project 2

Source:
1) Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin: Attention Is All You Need. CoRR abs/1706.03762 (2017)
2) https://jalammar.github.io/illustrated-transformer/
3) https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf
4) https://jalammar.github.io/illustrated-gpt2/
5) https://github.com/karpathy/nanoGPT

- Professor : Dr Ahmed ElSayed
- Date : Dec 15, 2023

- Group 4:

Naaz Nagori(Nazrinbanu)
Linda(Pham,Nguyen Linh Dan)
Aayush Sahare

Project Steps:

- Transformer Understanding
- Applications of transformer
- Significance of Self attention and multihead
- GPT2 (Transformer-Decoder)
- NanoGPT model
- Nanogpt training code for lyrics generation
- Iterations vs Loss Plots
- Results
- Conclusion

Transformer Understanding

Motivation for the Transformer Architecture:

- Recurrent models pose limitations due to their inherently sequential nature, hindering parallelization at longer sequence lengths.
- Attention mechanisms have proven effective in modeling dependencies without considering their distance in sequences but are typically used alongside recurrent networks.

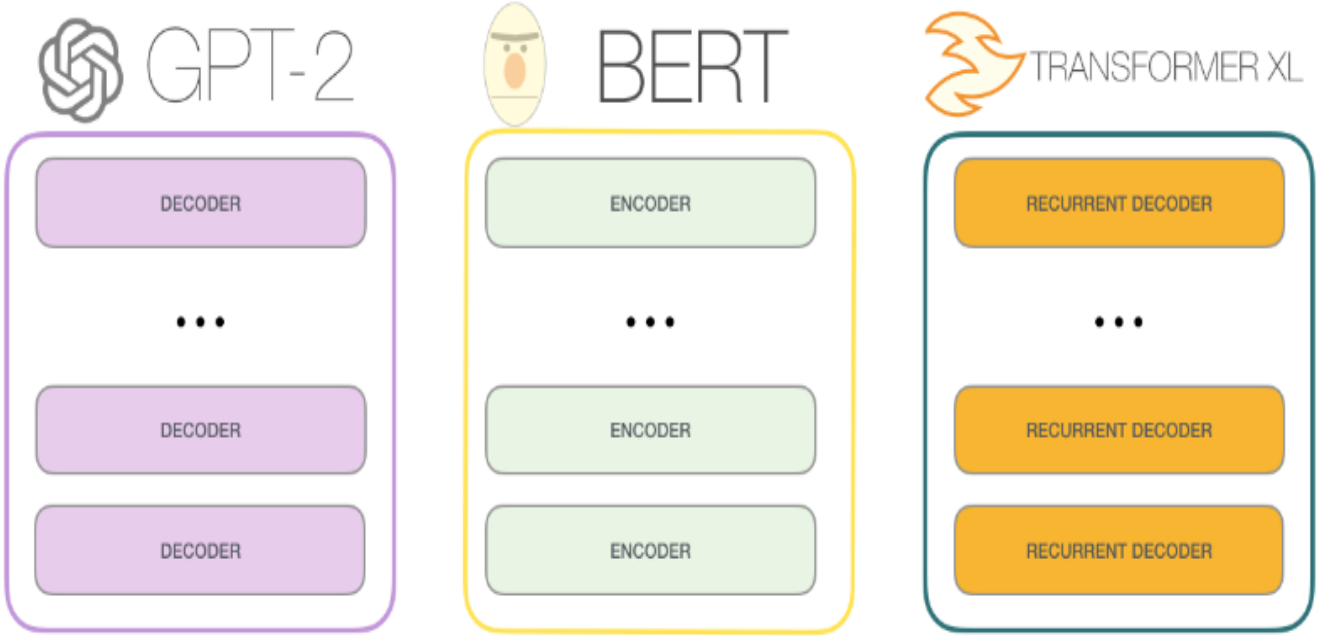
Usage of transformer

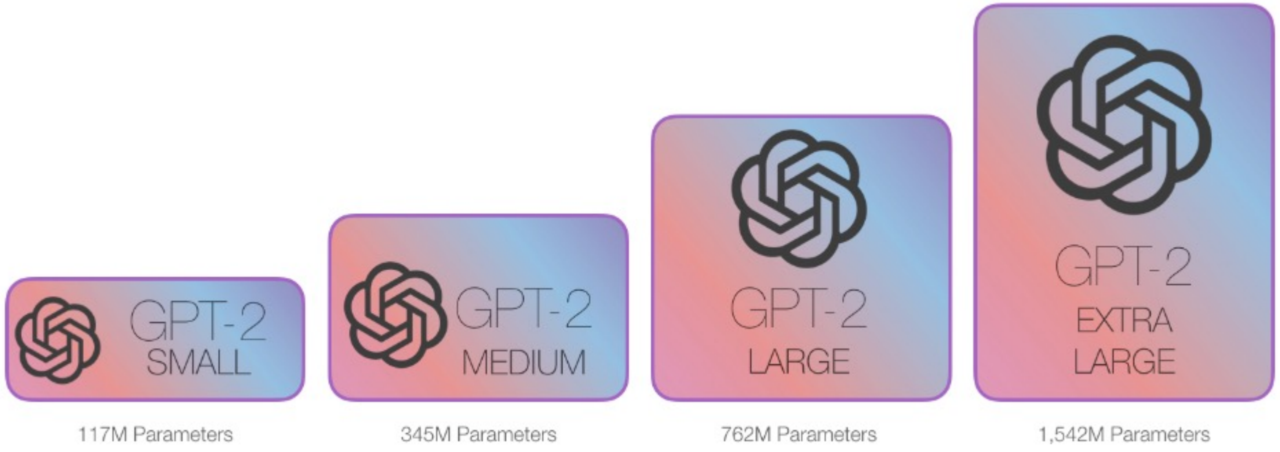
So what exactly is a language model?

What is a Language Model

In The Illustrated Word2vec, we've looked at what a language model is – basically a machine learning model that is able to look at part of a sentence and predict the next word. The most famous language models are smartphone keyboards that suggest the next word based on what you've currently typed.

In this sense, we can say that the GPT-2 is basically the next word prediction feature of a keyboard app, but one that is much larger and more sophisticated than what your phone has. The GPT-2 was trained on a massive 40GB dataset called WebText that the OpenAI researchers crawled from the internet as part of the research effort. To compare in terms of storage size, the keyboard app I use, SwiftKey, takes up 78MBs of space. The smallest variant of the trained GPT-2, takes up 500MBs of storage to store all of its parameters. The largest GPT-2 variant is 13 times the size so it could take up more than 6.5 GBs of storage space.





Source: https://jalammar.github.io/illustrated-gpt2/

Attention Is All You Need

by Google

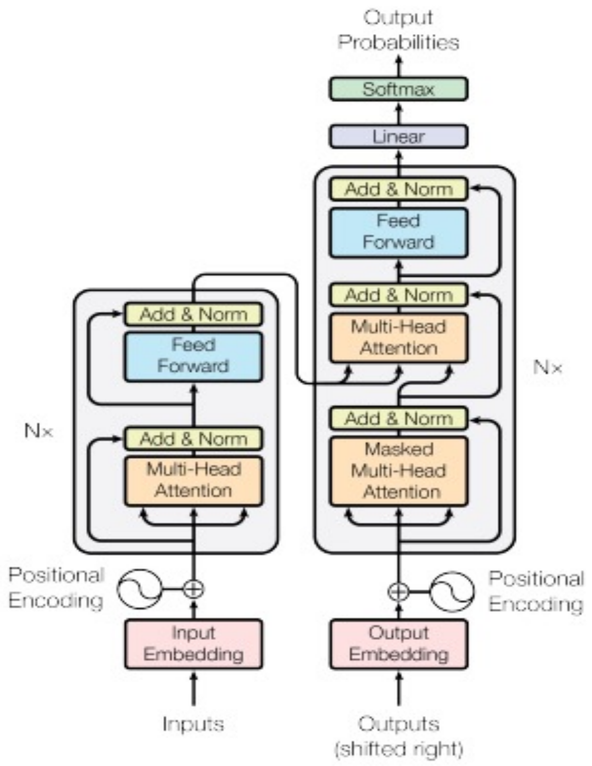


Figure 1: The Transformer - model architecture.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

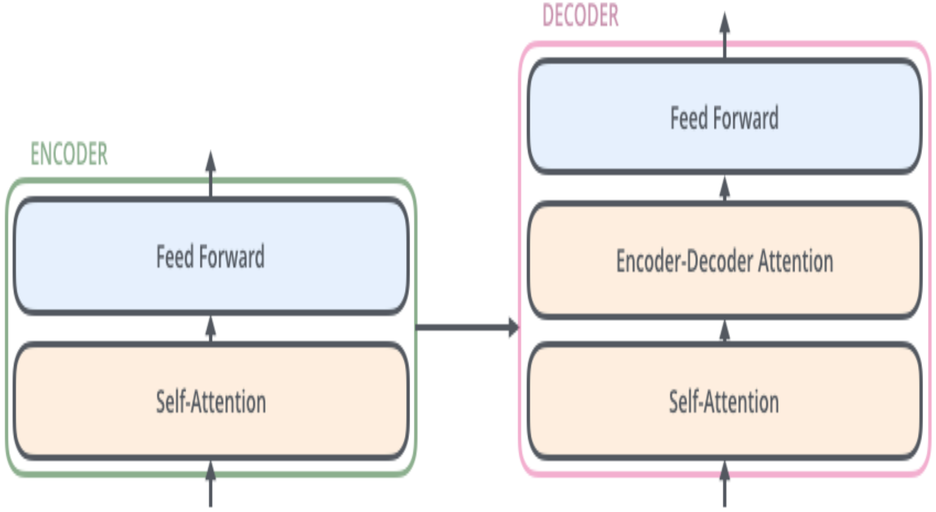
3.1 Encoder and Decoder Stacks

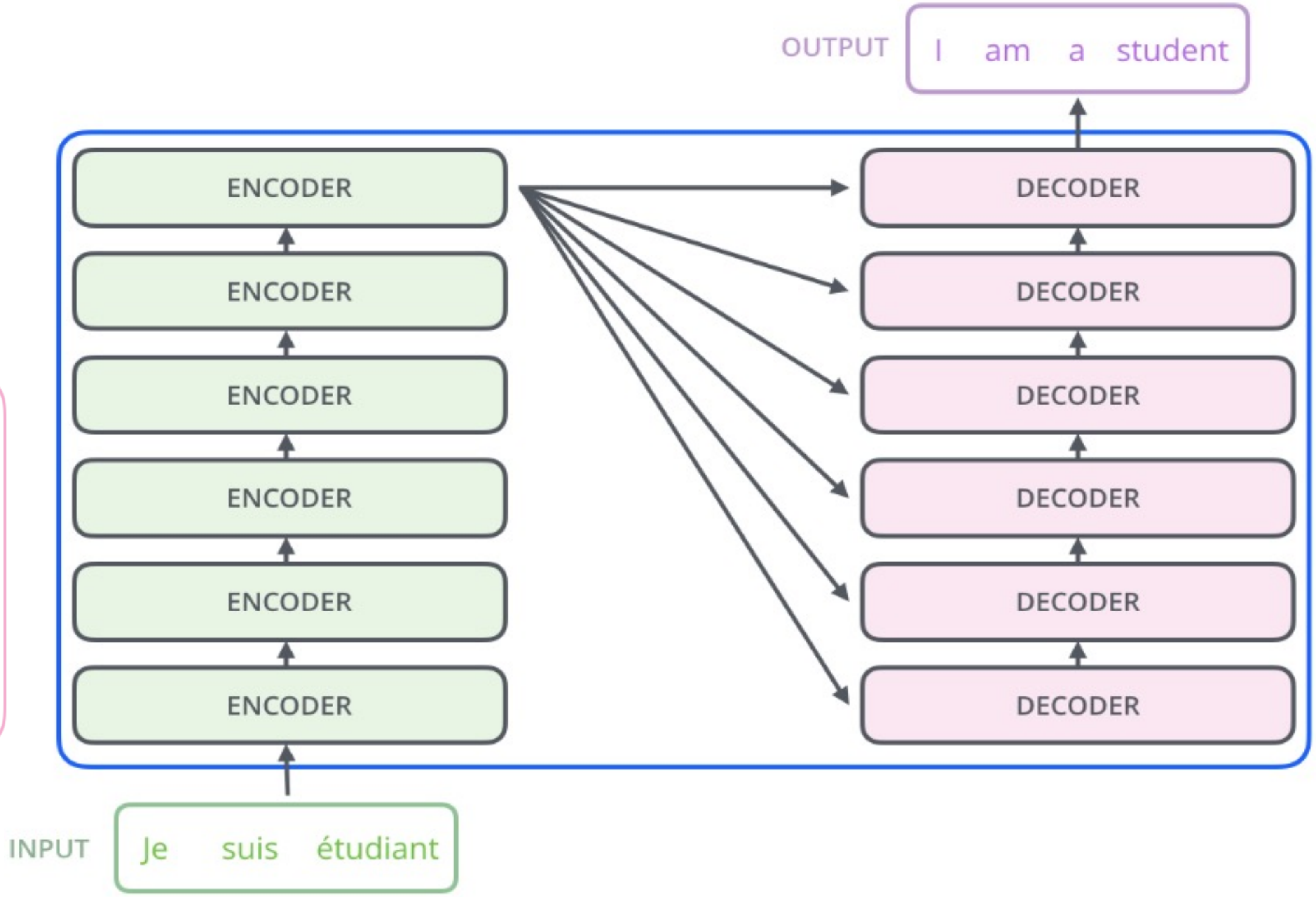
Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head

Transformer Understanding

- Transformer encoder-decoder architecture



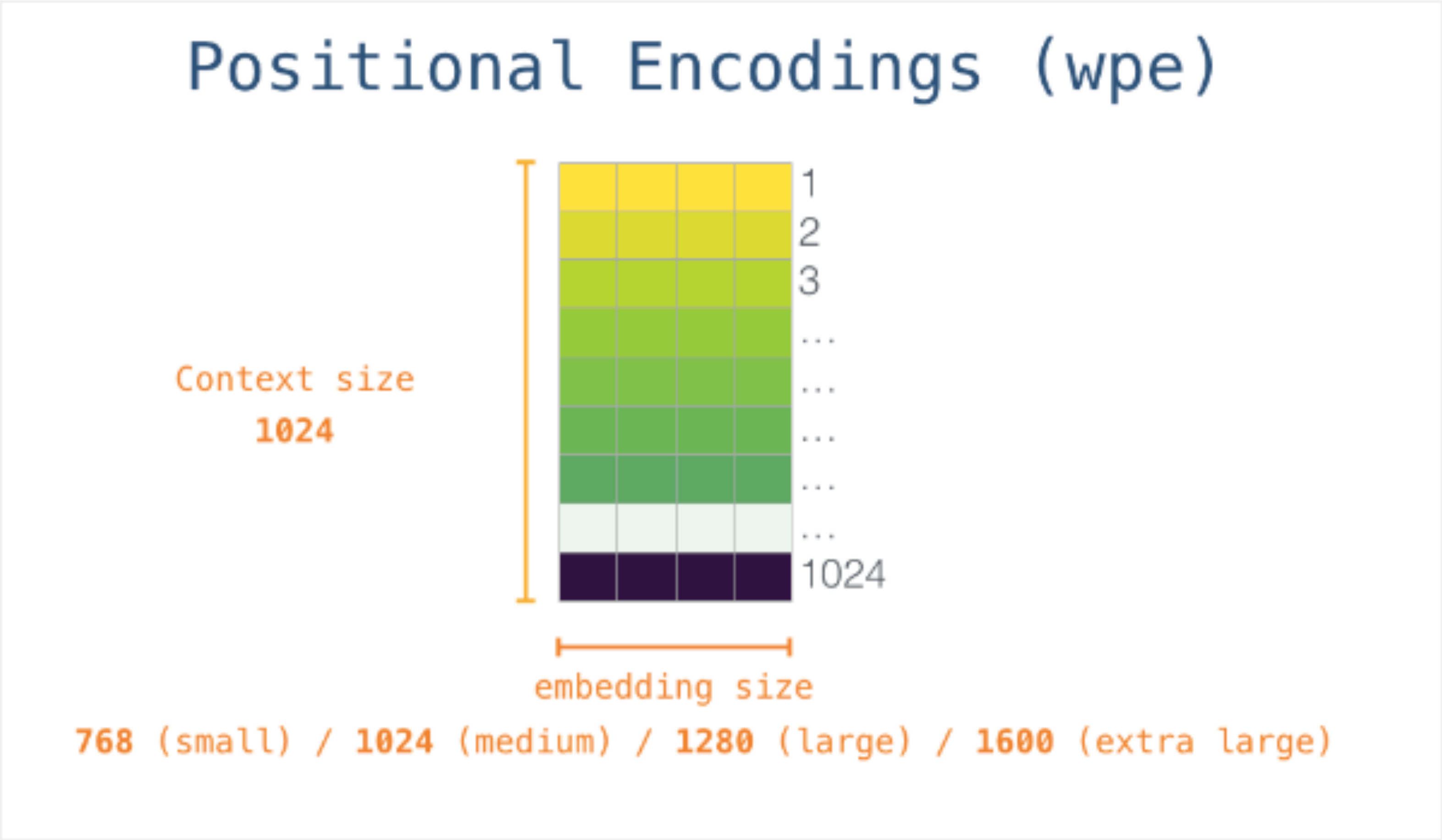


Transformer Model Architecture

- **Encoder and Decoder Stacks:**
  - **Encoder**: Comprises N = 6 identical layers, each consisting of two sub-layers:
     - Multi-head self-attention mechanism.
     - Position-wise fully connected feed-forward network.
     - Utilizes residual connections around each sub-layer, followed by layer normalization.
     - Produces outputs of dimension dmodel = 512 for all sub-layers and embedding layers.
  - **Decoder**: Also comprises N = 6 identical layers. Each layer contains three sub-layers:
     - Multi-head self-attention.
     - Another multi-head attention mechanism over the encoder stack's output.
     - Position-wise fully connected feed-forward network.
     - Similar residual connections and layer normalization as the encoder.

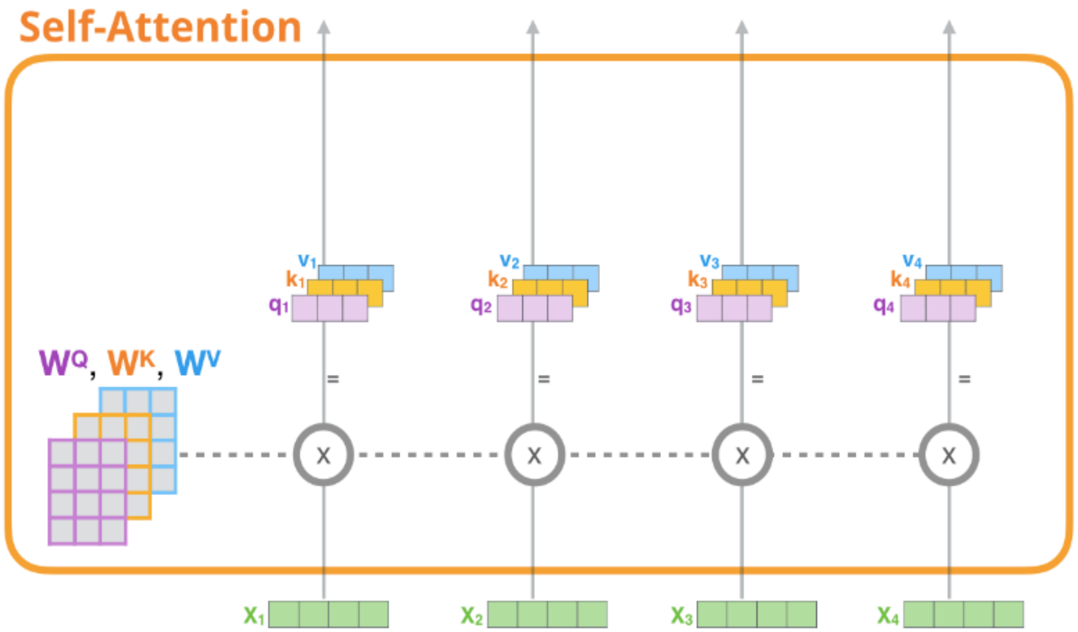
- **Attention Mechanisms:**
  - **Scaled Dot-Product Attention:**
     - Utilizes queries, keys, and values of dimensions dk and dv.
     - Computes dot products of queries and keys, followed by softmax to obtain weights on values.
     - Computes attention function on sets of queries packed into matrices.
     - Prevents extremely large gradients by scaling dot products.
  - **Multi-Head Attention:**
     - Linearly projects queries, keys, and values h times with learned linear projections to different dimensions.
     - Performs attention function in parallel, allowing joint attention to different representation subspaces.
     - Employs h = 8 parallel attention layers with dk = dv = dmodel/h = 64.

Positional Embeddings

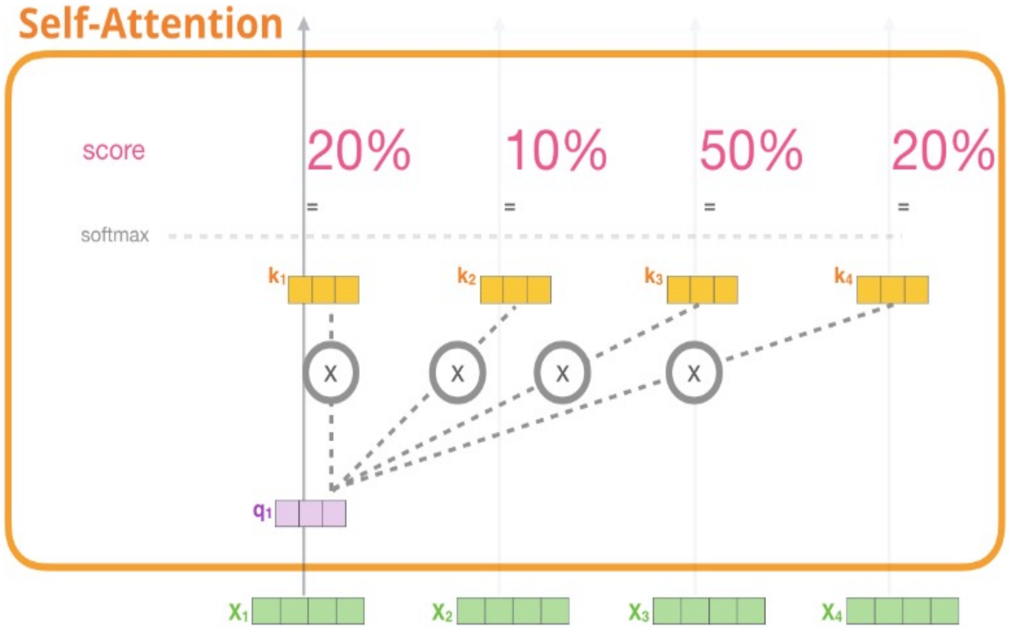


Self attention

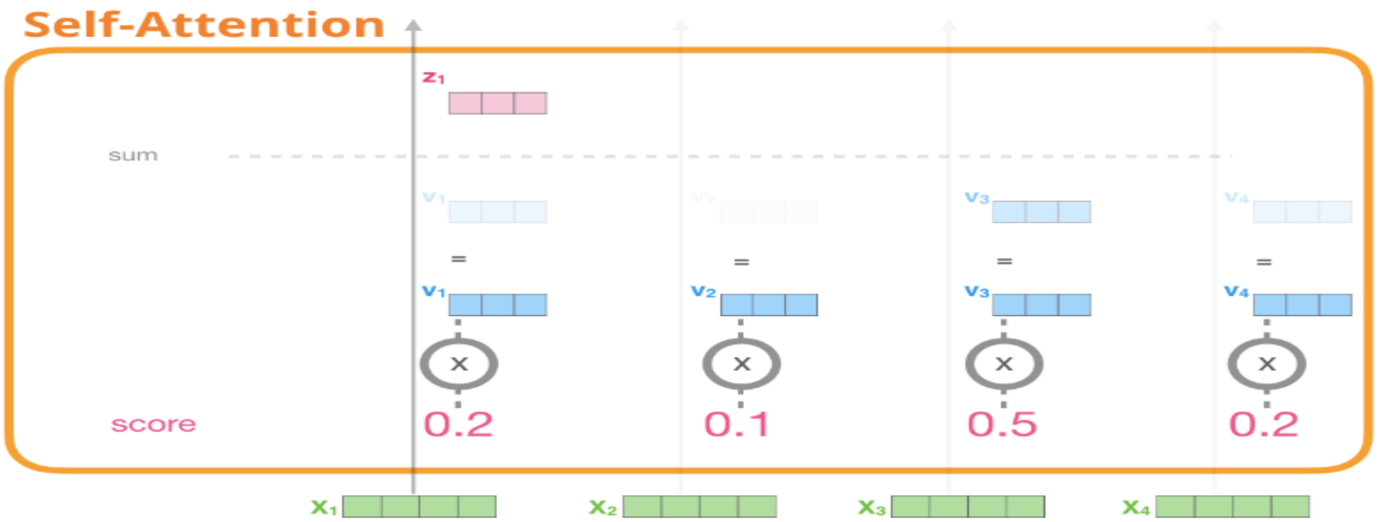
1) For each input token, create a query vector, a key vector, and a value vector by multiplying by weight Matrices W^Q, W^K, W^V



2) Multiply (dot product) the current query vector, by all the key vectors, to get a score of how well they match



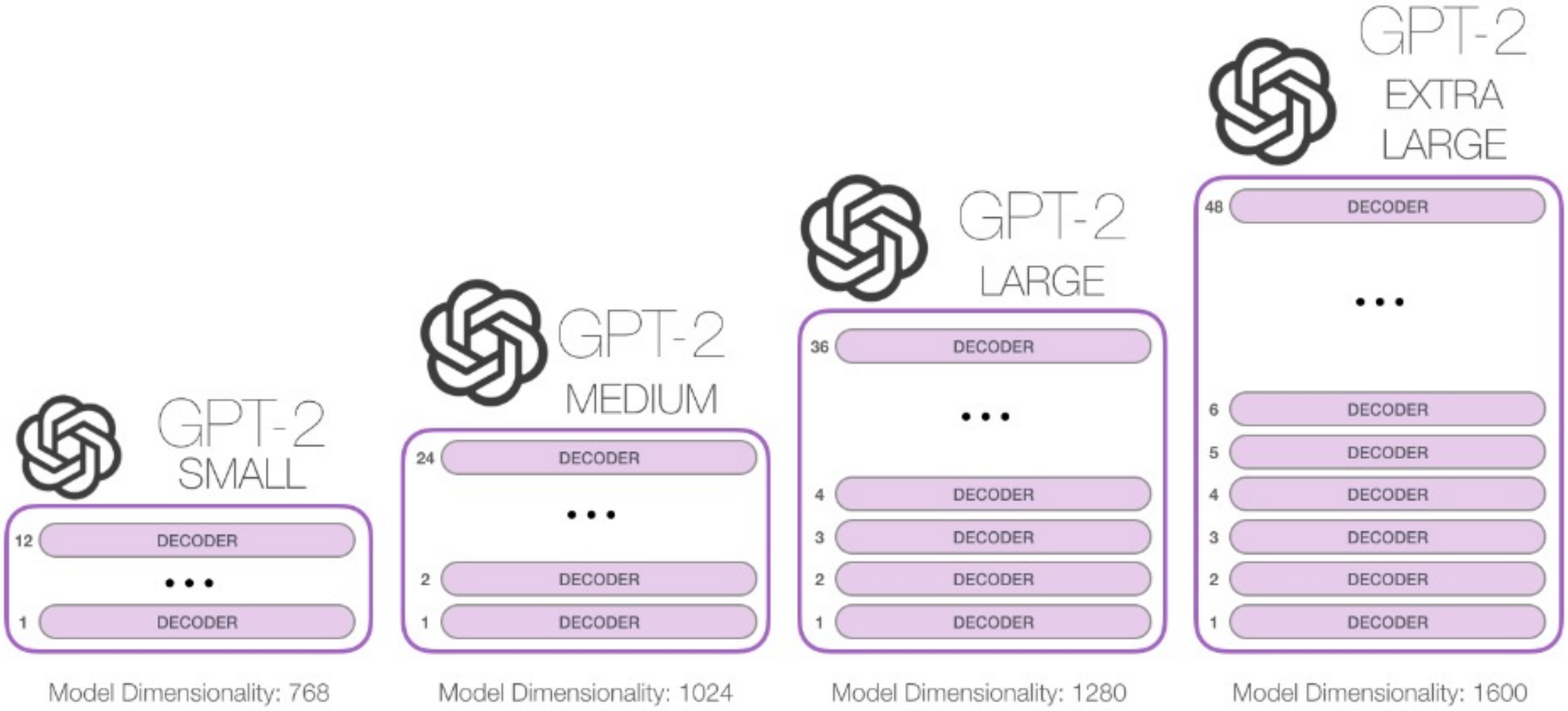
3) Multiply the value vectors by the scores, then sum up



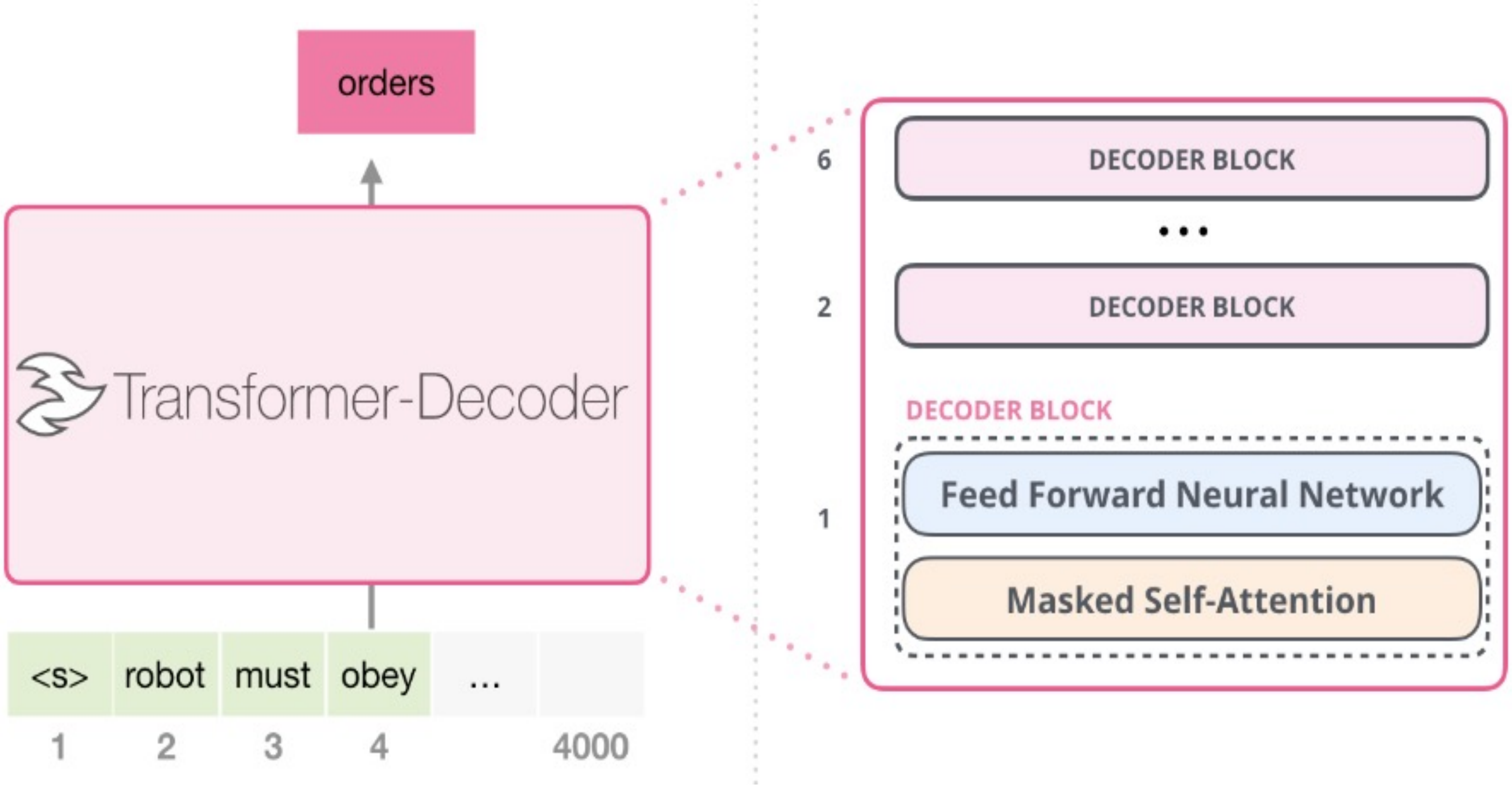
$$\mathrm{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

GPT2 Architecture

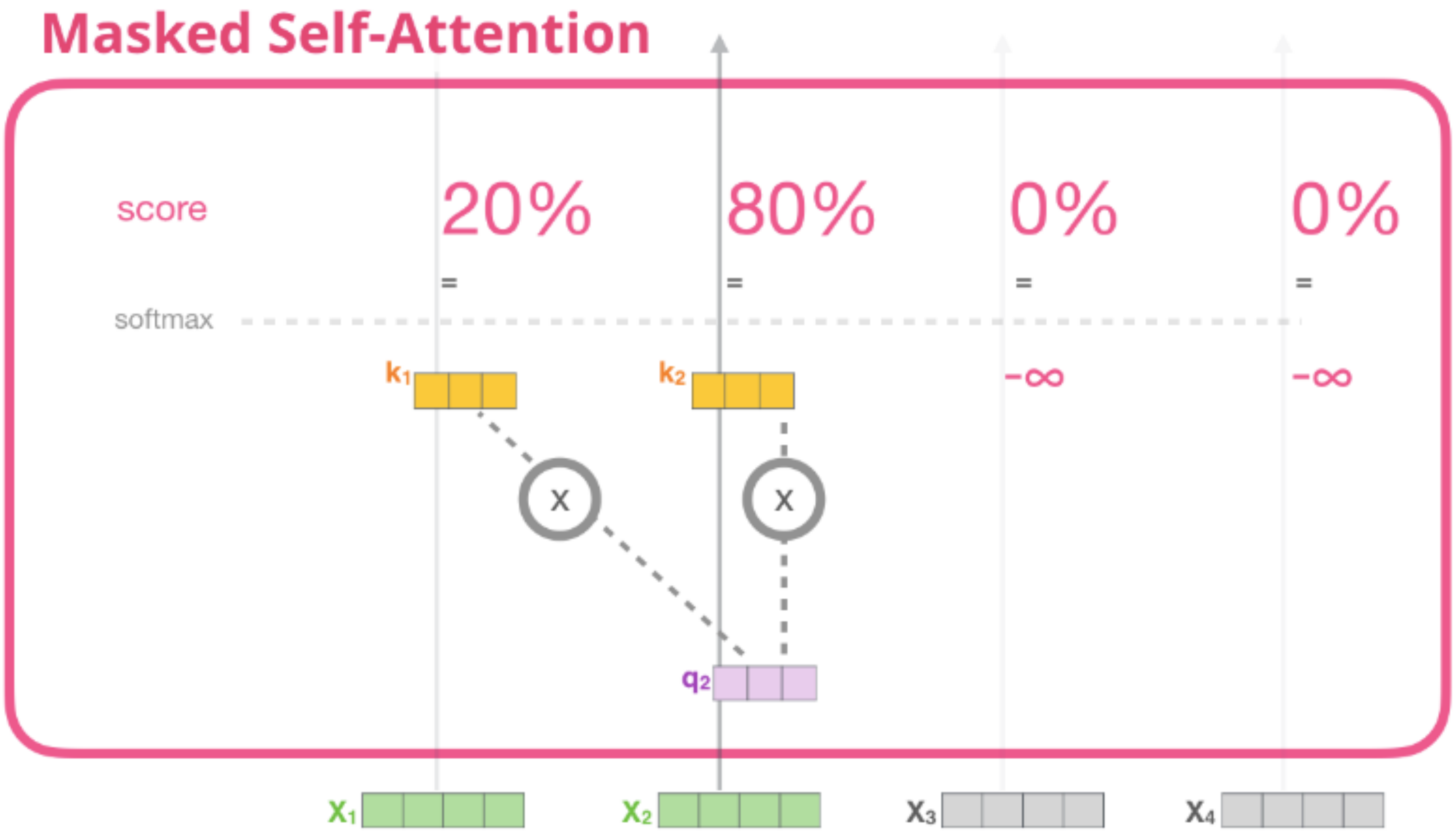
GPT2 Architecture



GPT2 Architecture



GPT2 Architecture



NanoGPT model

Train_lyrics.py

```python
# train a miniature nanoGPT model
# good for debugging and playing on macbooks and such

out_dir = 'out-lyrics'
eval_interval = 250 # keep frequent because we'll overfit
eval_iters = 200
log_interval = 10 # don't print too too often

# we expect to overfit on this small dataset, so only save when val improves
always_save_checkpoint = False

wandb_log = False # override via command line if you like
wandb_project = 'lyrics'
wandb_run_name = 'mini-gpt'

dataset = 'lyrics'
gradient_accumulation_steps = 1
batch_size = 64
block_size = 256 # context of up to 256 previous characters

# baby GPT model :)
n_layer = 6
n_head = 6
n_embd = 384
dropout = 0.2

learning_rate = 1e-3 # with baby networks can afford to go a bit higher
max_iters = 5000
lr_decay_iters = 5000 # make equal to max_iters usually
min_lr = 1e-4 # learning_rate / 10 usually
beta2 = 0.99 # make a bit bigger because number of tokens per iter is small

eval_interval = 10
warmup_iters = 100 # not super necessary potentially

# on macbook also add
# device = 'cpu'  # run on cpu only
# compile = False # do not torch compile the model
```

Train Dataset

Read 'text' field from spotify_millsongdata.csv file for training nanoGPT

Text Dataset

spotify_millsongdata

artist	song	link	text
ABBA	Ahe's My Kind Of Girl	/a/abba/ahes+my+kind+of+girl_20598417.html	Look at her face, it's a wonderful face And it means something special to me Look at the way that she smiles when she sees me How lucky can one fellow be? She's just my kind of girl, she makes me feel fine Who could ever believe that she could be mine? She's just my kind of girl, without her I'm blue And if she ever leaves me what could I do, what could I do? And when we go for a walk in the park
ABBA	Andante, Andante	/a/abba/andante+andante_20002708.html	Take it easy with me, please Touch me gently like a summer evening breeze Take your time, make it slow Andante, Andante Just let the feeling grow Make your fingers soft and light Let your body be the velvet of the night Touch my soul, you know how Andante, Andante Go slowly with me now I'm your music (I am your music and I am your song) I'm your song (I am your music and I am your song) Play me time and time again and make me strong (Play me again 'cause you're making me strong) Make me sing, make me sound

Nanogpt training data for lyrics generation

Prepare.py (Train/Val data)

- **Importing Libraries:**
- **Reading Data**
- **Data Splitting**
- **Text Encoding with GPT-2 BPE**
- **Output Preparatios**

```
import os
import tiktoken
import numpy as np
import pandas as pd

df = pd.read_csv('data/lyrics/spotify_millsongdata.csv')
data = df['text'].str.cat(sep='\n')

n = len(data)
train_data = data[:int(n*0.9)]
val_data = data[int(n*0.9):]

# encode with tiktoken gpt2 bpe
enc = tiktoken.get_encoding("gpt2")
train_ids = enc.encode_ordinary(train_data)
val_ids = enc.encode_ordinary(val_data)
print(f"train has {len(train_ids):,} tokens")
print(f"val has {len(val_ids):,} tokens")

# export to bin files
train_ids = np.array(train_ids, dtype=np.uint16)
val_ids = np.array(val_ids, dtype=np.uint16)
train_ids.tofile(os.path.join(os.path.dirname(__file__), 'train.bin'))
val_ids.tofile(os.path.join(os.path.dirname(__file__), 'val.bin'))

# train.bin has 301,966 tokens
# val.bin has 36,059 tokens
```

Device and Hyper Parameters

Device setting: Bring all tensors to device which is set to CPU at this time

- The GPT-2 model is initialized based on the specified configuration.
- The script supports initializing the model from scratch, resuming training from a checkpoint, or using pre-trained GPT-2 weights.

```
!python /content/nanoGPT/train.py /content/nanoGPT/config/train_lyrics.py --device=cuda --compile=False --eval_iters=10 --log_interval=1 --block_size=64 --batch_size=12 --n_layer=4 --n_head=4 --n_embd=128 --max_iters=1000 --lr_decay_iters=2000 --dropout=0.0
```

```
val has 2,456,916 tokens
```

```
1 !python /content/nanoGPT/train.py /content/nanoGPT/config/train_lyrics.py
```

```
warmup_iters = 100 # not super necessary potentially

# on macbook also add
# device = 'cpu'  # run on cpu only
# compile = False # do not torch compile the model
Overriding: device = cuda
Overriding: compile = False
Overriding: eval_iters = 20
Overriding: log_interval = 1
Overriding: block_size = 64
Overriding: batch_size = 12
Overriding: n_layer = 4
Overriding: n_head = 4
Overriding: n_embd = 128
Overriding: max_iters = 1000
Overriding: lr_decay_iters = 2000
Overriding: dropout = 0.0
tokens per iteration will be: 768
Initializing a new model from scratch
defaulting to vocab_size of GPT-2 to 50304 (50257 rounded up for efficiency)
number of parameters: 7.23M
num decayed parameter tensors: 18, with 7,233,536 parameters
num non-decayed parameter tensors: 9, with 1,152 parameters
using fused AdamW: True
step 0: train loss 10.7153, val loss 10.7146
iter 0: loss 10.6806, time 1746.56ms, mfu -100.00%
iter 1: loss 10.6810, time 10.36ms, mfu -100.00%
iter 2: loss 10.6791, time 24.31ms, mfu -100.00%
iter 3: loss 10.6747, time 25.95ms, mfu -100.00%
iter 4: loss 10.5875, time 26.42ms, mfu -100.00%
iter 5: loss 10.4791, time 26.05ms, mfu 0.41%
iter 6: loss 10.4510, time 26.17ms, mfu 0.41%
iter 7: loss 10.4392, time 26.73ms, mfu 0.41%
iter 8: loss 10.2845, time 26.03ms, mfu 0.41%
iter 9: loss 10.2997, time 26.05ms, mfu 0.41%
iter 10: loss 10.1449, time 25.72ms, mfu 0.41%
iter 11: loss 10.1541, time 29.38ms, mfu 0.41%
iter 12: loss 10.0913, time 17.48ms, mfu 0.43%
```

batchsize * block size

Iterations vs Loss Plots

+ Code + Text

@250

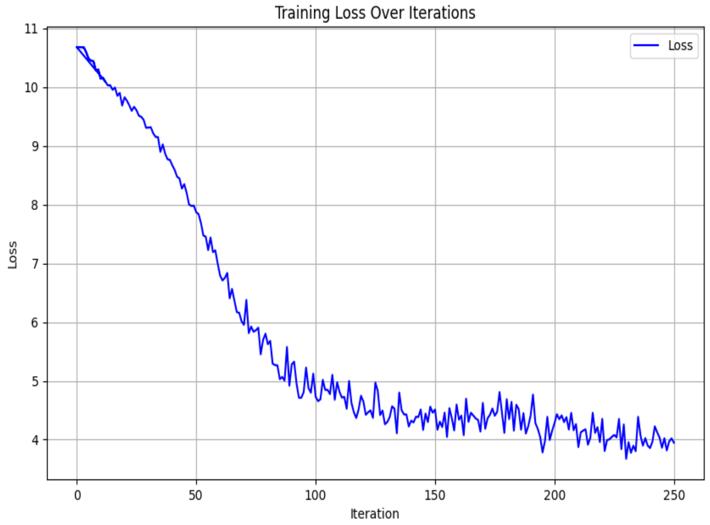
```
iter 230: loss 3.6741, time 1781.08ms, mfu 0.01%
iter 231: loss 3.9546, time 1778.10ms, mfu 0.01%
iter 232: loss 3.7749, time 1593.56ms, mfu 0.01%
iter 233: loss 3.8953, time 1172.00ms, mfu 0.01%
iter 234: loss 3.8030, time 1193.55ms, mfu 0.01%
iter 235: loss 4.3871, time 1198.96ms, mfu 0.01%
iter 236: loss 4.0717, time 1208.16ms, mfu 0.01%
iter 237: loss 3.8979, time 1259.43ms, mfu 0.01%
iter 238: loss 4.0242, time 1189.43ms, mfu 0.01%
iter 239: loss 3.8989, time 1218.08ms, mfu 0.01%
iter 240: loss 3.8564, time 1187.60ms, mfu 0.01%
iter 241: loss 3.9642, time 1700.38ms, mfu 0.01%
iter 242: loss 4.2251, time 1806.61ms, mfu 0.01%
iter 243: loss 4.1229, time 1777.88ms, mfu 0.01%
iter 244: loss 4.0324, time 1736.13ms, mfu 0.01%
iter 245: loss 3.8615, time 1627.83ms, mfu 0.01%
iter 246: loss 4.0237, time 1193.44ms, mfu 0.01%
iter 247: loss 3.8156, time 1168.97ms, mfu 0.01%
iter 248: loss 3.9676, time 1161.53ms, mfu 0.01%
iter 249: loss 4.0204, time 1156.56ms, mfu 0.01%
step 250: train loss 4.0635, val loss 4.1174
saving checkpoint to out-lyrics
iter 250: loss 3.9478, time 21959.95ms, mfu 0.01%
```

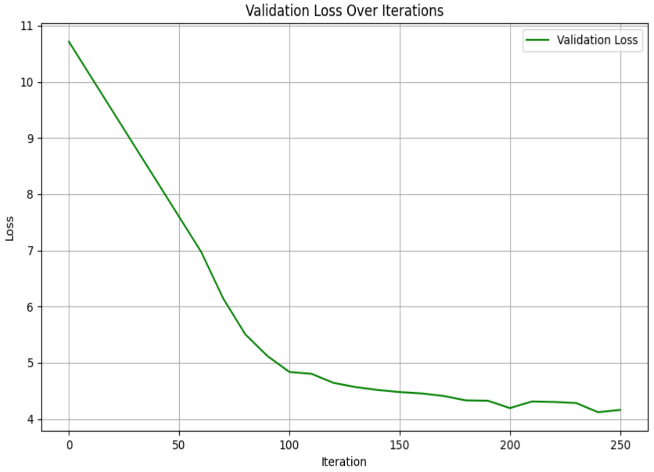
@1000

```
iter 977: loss 3.6164, time 21.74ms, mfu 0.51%
iter 978: loss 3.5689, time 20.57ms, mfu 0.51%
iter 979: loss 3.4867, time 21.06ms, mfu 0.51%
iter 980: loss 4.1370, time 22.67ms, mfu 0.50%
iter 981: loss 3.6086, time 19.46ms, mfu 0.51%
iter 982: loss 3.7003, time 20.83ms, mfu 0.51%
iter 983: loss 3.2515, time 20.84ms, mfu 0.51%
iter 984: loss 3.5606, time 21.68ms, mfu 0.51%
iter 985: loss 3.5288, time 22.14ms, mfu 0.51%
iter 986: loss 3.8286, time 22.39ms, mfu 0.50%
iter 987: loss 3.3172, time 21.57ms, mfu 0.50%
iter 988: loss 3.7051, time 21.98ms, mfu 0.50%
iter 989: loss 3.3790, time 19.57ms, mfu 0.51%
iter 990: loss 3.5177, time 20.79ms, mfu 0.51%
iter 991: loss 3.5524, time 21.09ms, mfu 0.51%
iter 992: loss 3.9999, time 20.78ms, mfu 0.51%
iter 993: loss 3.5249, time 21.01ms, mfu 0.51%
iter 994: loss 3.6793, time 21.42ms, mfu 0.51%
iter 995: loss 3.6419, time 21.17ms, mfu 0.51%
iter 996: loss 3.7930, time 20.62ms, mfu 0.51%
iter 997: loss 3.8286, time 20.76ms, mfu 0.51%
iter 998: loss 4.2096, time 23.52ms, mfu 0.51%
iter 999: loss 3.5168, time 19.28ms, mfu 0.51%
step 1000: train loss 3.5402, val loss 3.7532
iter 1000: loss 3.6652, time 320.37ms, mfu 0.46%
```

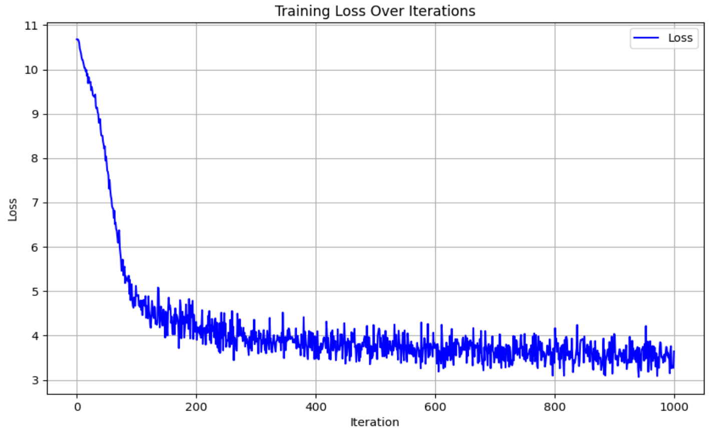
Code : Training Epochs vs Loss

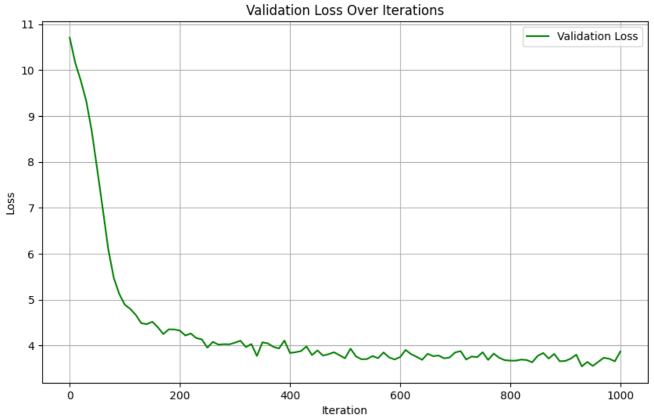
Train epoch = 250 iterations





Train epoch = 1000 iterations





Results@250 iterations

@10

```
1  !python /content/nanoGPT/sample.py --out_di
Overriding: device = cpu
NAAZ RESUME---------------
number of parameters: 7.23M
NAAZ init from a given GPT-2 model--------------
No meta.pkl found, assuming GPT-2 encodings...

A your way

I do like with it
SCh will got the world's my hand on at the body

And I'm a out

You find,
I'm the life

I don't love my way,
[I'll just know you fall
You's I one

That
She's be is my future
Your will you
G out't have my heart
Like, no know you could be
The heart


'll'm the this me


You'm as
I's her
He's be

I it's be
The young
The's a head
There's the mind
She's a world
I want
I'm to be,
But I're know to get the' tonight
She for to to want me in a more, that and you it

And it
```

@250

```
1  !python /content/nanoGPT/sample.py --out_dir
Overriding: out_dir = out-lyrics
Overriding: device = cpu
RESUME---------------
number of parameters: 7.23M
mata path--------------- data/lyrics/meta.pkl
No meta.pkl found, assuming GPT-2 encodings...
start='\n'


She it
H like with the heart
I couldn't believe's right you on all the reason
And I like a out
I're the find,
When I don't know
M the face
I're nothing
[I'll face
Oh I'll only I one on you was the me

She at that is my future
And you I could you
I see,

 could do
'm never there




And you I'd no there his be as
They a live's no a look me,
I live a love
I will be that I going,

I been me
.


The go, time
I like my love, the my matter
```

@1000

```
A good, what a time
When it like so only love
But you couldn't believe you were you on
They're just the breath
And I'm the love the morning as all,
Of all how you

I don't love my soul, I can't be
Oh, it'll do
I'm gonna hear (I'm on
And you can't baby
I feel will never say I could you
I'm no good
That know I could do
I don't want to tell you


I'm gonna have to know there
I got you before
It's waiting,
I'm gonna be and I can't want to me
But the love you never've been been
I don't you know where you ain't
The other day
I've like you
I'm gonna be in your love
And if you got for you
If you so that I will leave you want to know
I've just better

I've been you
I'm where I'm one
'
You're so I've been another
I was been in

I'm so love
I'm a fine
```

Conclusion:

- Tested with network trained with nanoGPT - writes a song
- Training loss and validation loss plot shows that after 250 iterations, validation

Loss is not reducing

- Our Model is nano/baby model, it might not perform as good as full fledge version but its still pretty good with 1 minute of training!

THANK YOU