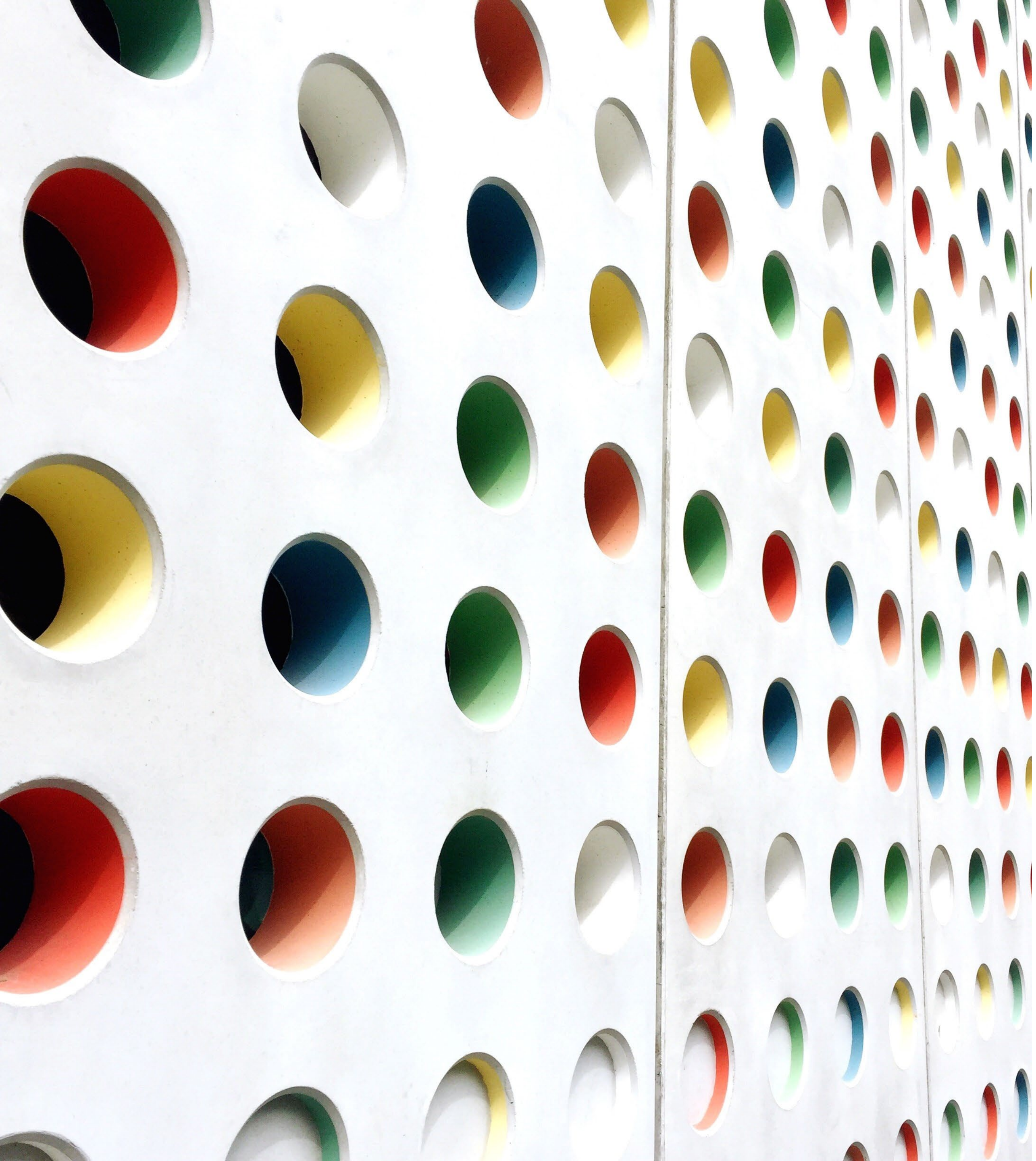


minGPT

HOW IT WORKS

DAVE BROWN
SIGNALPOP LLC
WWW.SIGNALPOP.COM

minGPT – Use Cases

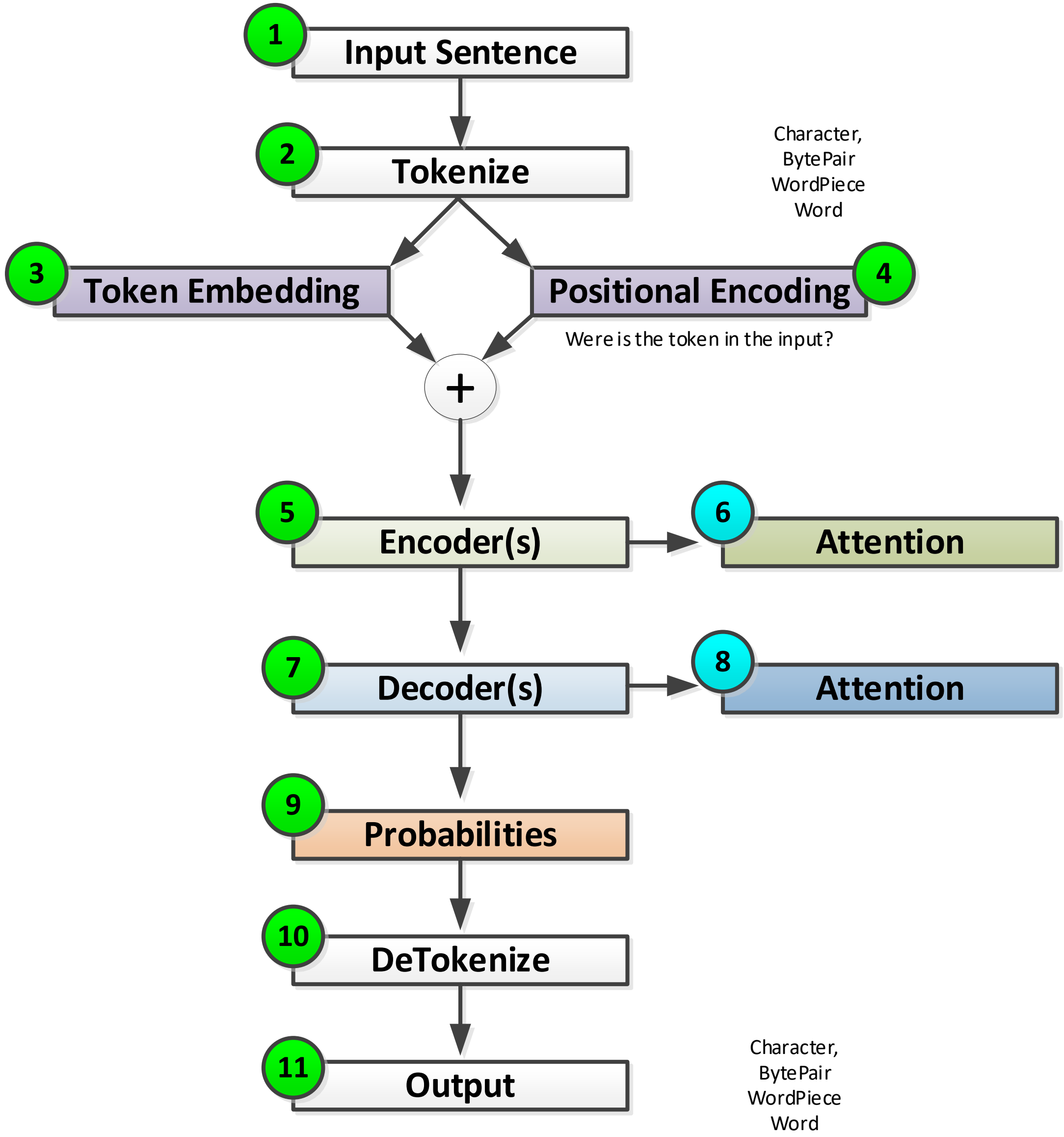
IMAGE CLASSIFICATION/CREATION

NATURAL LANGUAGE PROCESSING

TIME SERIES PREDICTIONS

General Transformer Model

1. Input sentence
2. Tokenize it (char, word, etc.)
3. Create token embedding and
4. Position encoding (e.g., sin(w))
5. Encoder creates encoding
6. Run self attention on encoding
7. Decode to create logits
8. Run self attention on logits
9. Softmax creates probabilities
10. Detokenize
11. Produce predicted next char/word.



minGPT

minGPT, created by Andrej Karpathy, is a simplified implementation of the original OpenAI GPT-2 open-source project.

GPT has proven very useful in solving many Natural Language Processing problems (NLP) and as shown by Karpathy and others, also used to solve tasks outside of the NLP domain such as generative image processing and classification.

This presentation seeks to visually show the overall design of Karpathy's minGPT implementation so as to better understand how it works.

For more information on minGPT, GPT, GPT-2 or ImageGPT, see the references at the end of this presentation.

As a last note, I wanted to send a big Thank You to Andrej Karpathy - your simplified implementations of complex models have always been an inspiration!

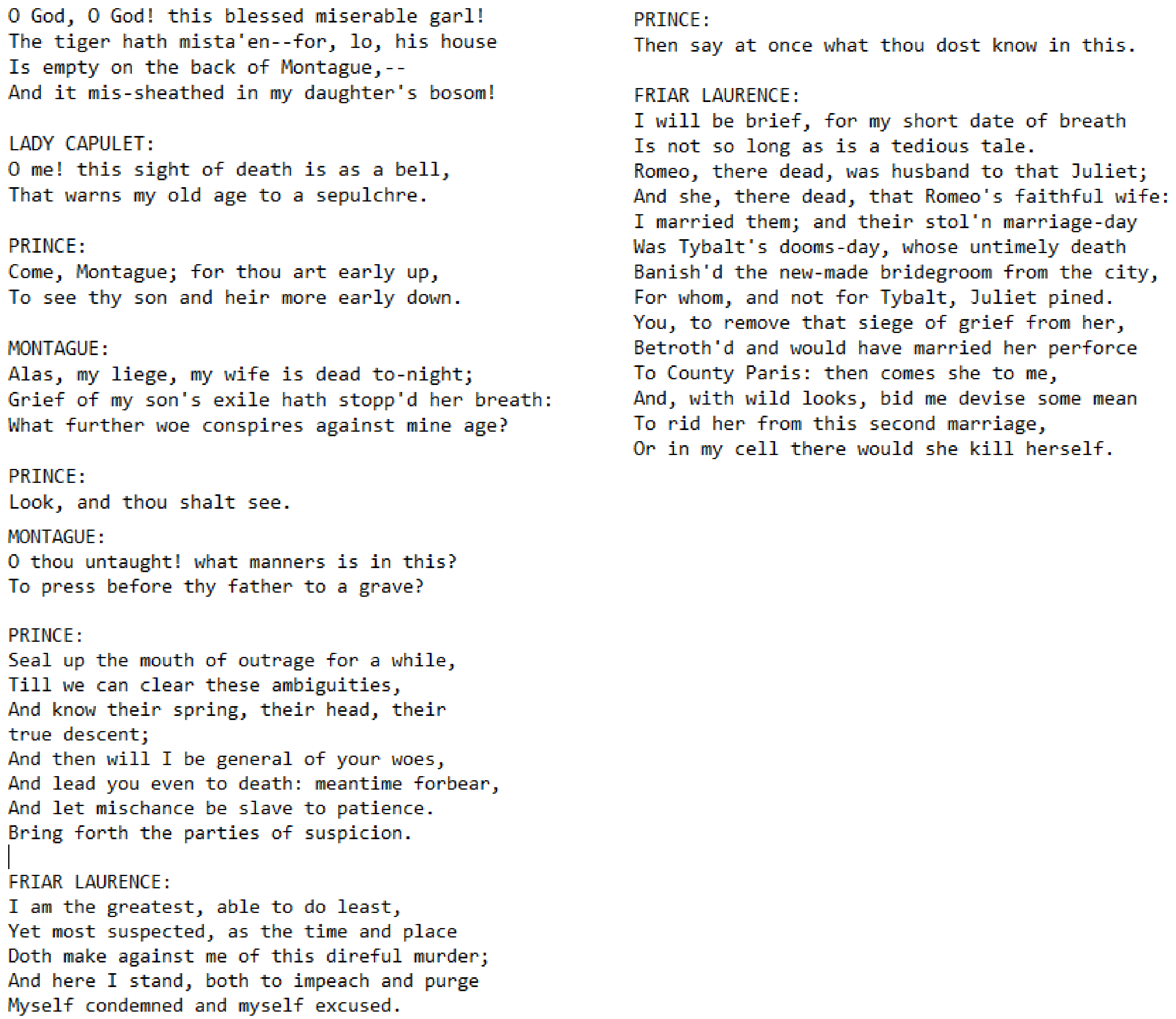
minGPT – Data Input (NLP)

Input:
“O God, O God!”

Run with

CasualSelfAttention

Output



```
O God, O God! this blessed miserable garl!
The tiger hath mista'en--for, lo, his house
Is empty on the back of Montague,--
And it mis-sheathed in my daughter's bosom!

LADY CAPULET:
O me! this sight of death is as a bell,
That warns my old age to a sepulchre.

PRINCE:
Come, Montague; for thou art early up,
To see thy son and heir more early down.

MONTAGUE:
Alas, my liege, my wife is dead to-night;
Grief of my son's exile hath stopp'd her breath:
What further woe conspires against mine age?

PRINCE:
Look, and thou shalt see.
MONTAGUE:
O thou untaught! what manners is in this?
To press before thy father to a grave?

PRINCE:
Seal up the mouth of outrage for a while,
Till we can clear these ambiguities,
And know their spring, their head, their
true descent;
And then will I be general of your woes,
And lead you even to death: meantime forbear,
And let mischance be slave to patience.
Bring forth the parties of suspicion.

FRIAR LAURENCE:
I am the greatest, able to do least,
Yet most suspected, as the time and place
Doth make against me of this direful murder;
And here I stand, both to impeach and purge
Myself condemned and myself excused.

PRINCE:
Then say at once what thou dost know in this.

FRIAR LAURENCE:
I will be brief, for my short date of breath
Is not so long as is a tedious tale.
Romeo, there dead, was husband to that Juliet;
And she, there dead, that Romeo's faithful wife:
I married them; and their stol'n marriage-day
Was Tybalt's dooms-day, whose untimely death
Banish'd the new-made bridegroom from the city,
For whom, and not for Tybalt, Juliet pined.
You, to remove that siege of grief from her,
Betroth'd and would have married her perforce
To County Paris: then comes she to me,
And, with wild looks, bid me devise some mean
To rid her from this second marriage,
Or in my cell there would she kill herself.
```

minGPT - Data Input (NLP)

Input:
"O God, O God!"

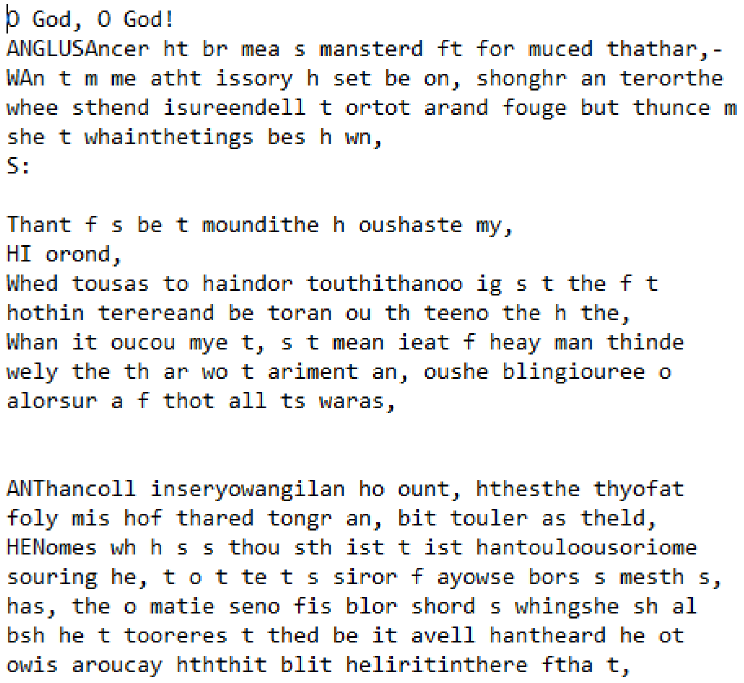
According to Karpathy, his implementation uses a "**totally unnecessary custom causal self-attention module**"

In this slide we put that to the test by removing it from the model.

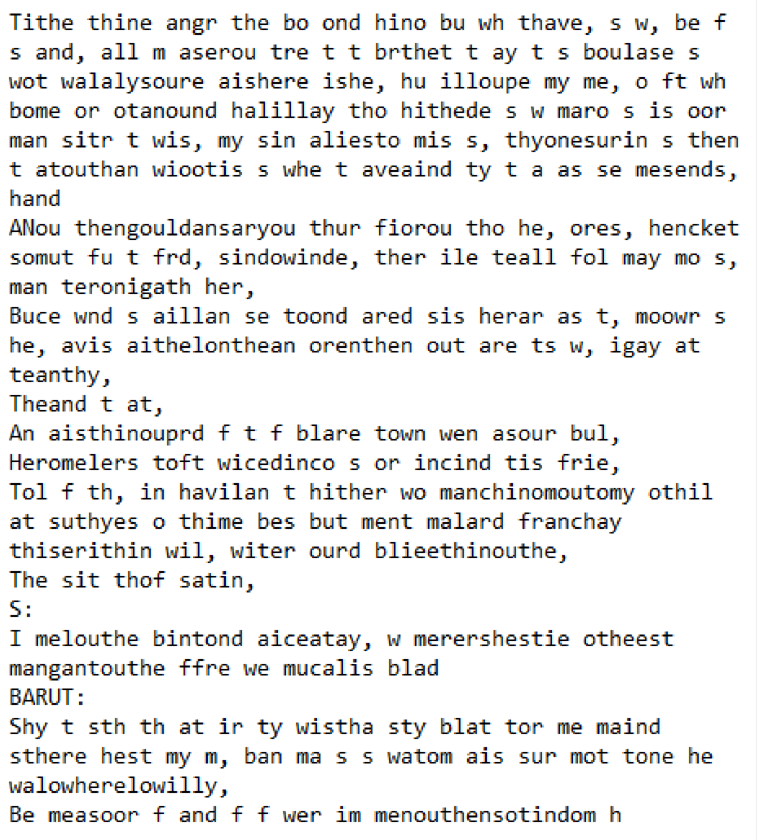
Run without

CasualSelfAttention

Output



```
O God, O God!
ANGLUSAncer ht br mea s mansterd ft for muced thathar,-
WAn t m me atht issory h set be on, shonghr an terorthe
whee sthend isureendell t ortot arand fouge but thunce m
she t whainthetings bes h wn,
S:

Thant f s be t moundithe h oushaste my,
HI orond,
Whed tousas to haindor touthithanoo ig s t the f t
hothin terereand be toran ou th teeno the h the,
Whan it oucou mye t, s t mean ieat f heay man thinde
wely the th ar wo t ariment an, oushe blingiouree o
alorsur a f thot all ts waras,


ANThancoll inseryowangilan ho ount, hthesthe thyofat
foly mis hof thared tongr an, bit touler as theld,
HENomes wh h s s thou sth ist t ist hantouloousoriome
souring he, t o t te t s siror f ayowse bors s mesth s,
has, the o matie seno fis blor shord s whingshe sh al
bsh he t tooreres t thed be it avell hantheard he ot
owis aroucay hththit blit heliritinthere ftha t,
```



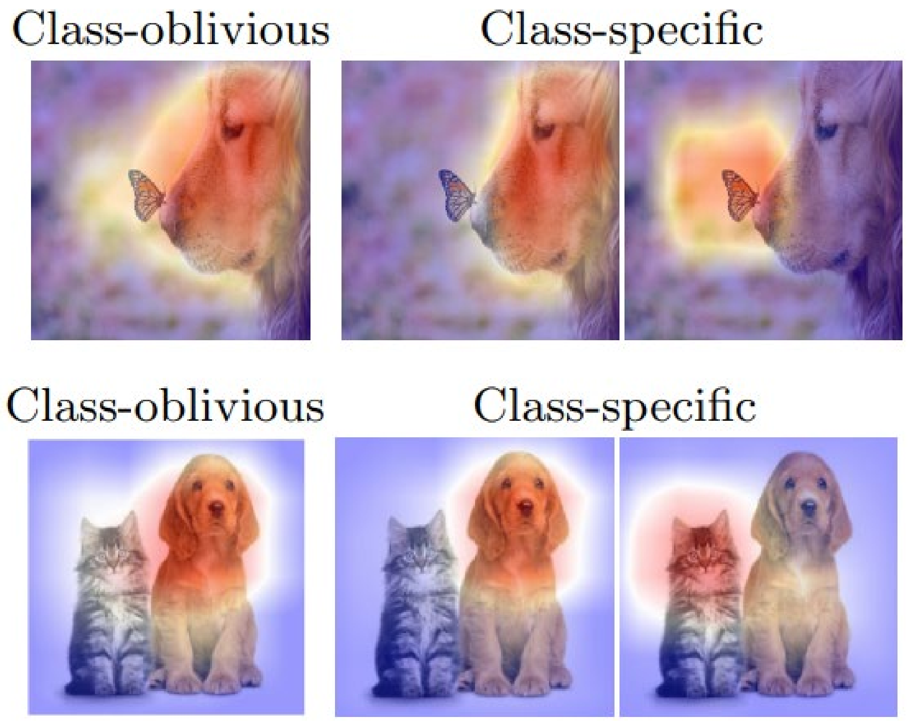
```
Tithe thine angr the bo ond hino bu wh thave, s w, be f
s and, all m aserou tre t t brthet t ay t s boulase s
wot walalysoure aishere ishe, hu illoupe my me, o ft wh
bome or otanound halillay tho hithede s w maro s is oor
man sitr t wis, my sin aliesto mis s, thyonesurin s then
t atouthan wiootis s whe t aveaind ty t a as se mesends,
hand
ANou thengouldansaryou thur fiorou tho he, ores, hencket
somut fu t frd, sindowinde, ther ile teall fol may mo s,
man teronigath her,
Buce wnd s aillan se toond ared sis herar as t, moowr s
he, avis aithelonthean orenthen out are ts w, igay at
teanthy,
Theand t at,
An aisthinouprd f t f blare town wen asour bul,
Heromelers toft wicedinco s or incind tis frie,
Tol f th, in havilan t hither wo manchinomoutomy othil
at suthyes o thime bes but ment malard franchay
thiserithin wil, witer ourd blieethinouthe,
The sit thof satin,
S:
I melouthe bintond aiceatay, w merershestie otheest
mangantouthe ffre we mucalis blad
BARUT:
Shy t sth th at ir ty wistha sty blat tor me maind
sthere hest my m, ban ma s s watom ais sur mot tone he
walowherelowilly,
Be measoor f and f f wer im menouthensotindom h
```

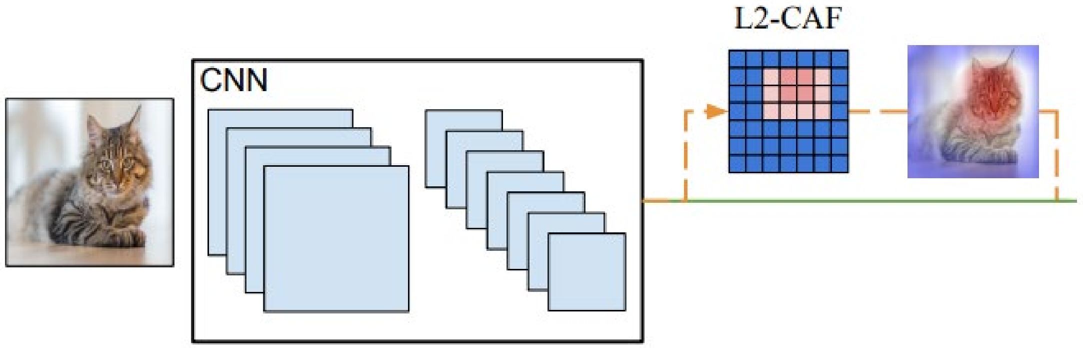
From our results as shown above, the CasualSelfAttention seems important!

What is Attention?

Attention focuses on the important aspects of the input – for example, visual attention focuses on important aspect of image.

(see Taha reference for more details)





Why is Attention Important?

Attention focuses on the important tokens and gives them context.

(see Wood reference for more details)

dk = dimension of keys, a design time hyper parameter.
Q = vector of queries, dk in dimension.
K = vector of keys, dk in dimension.
V = vector of values, dk in dimension.

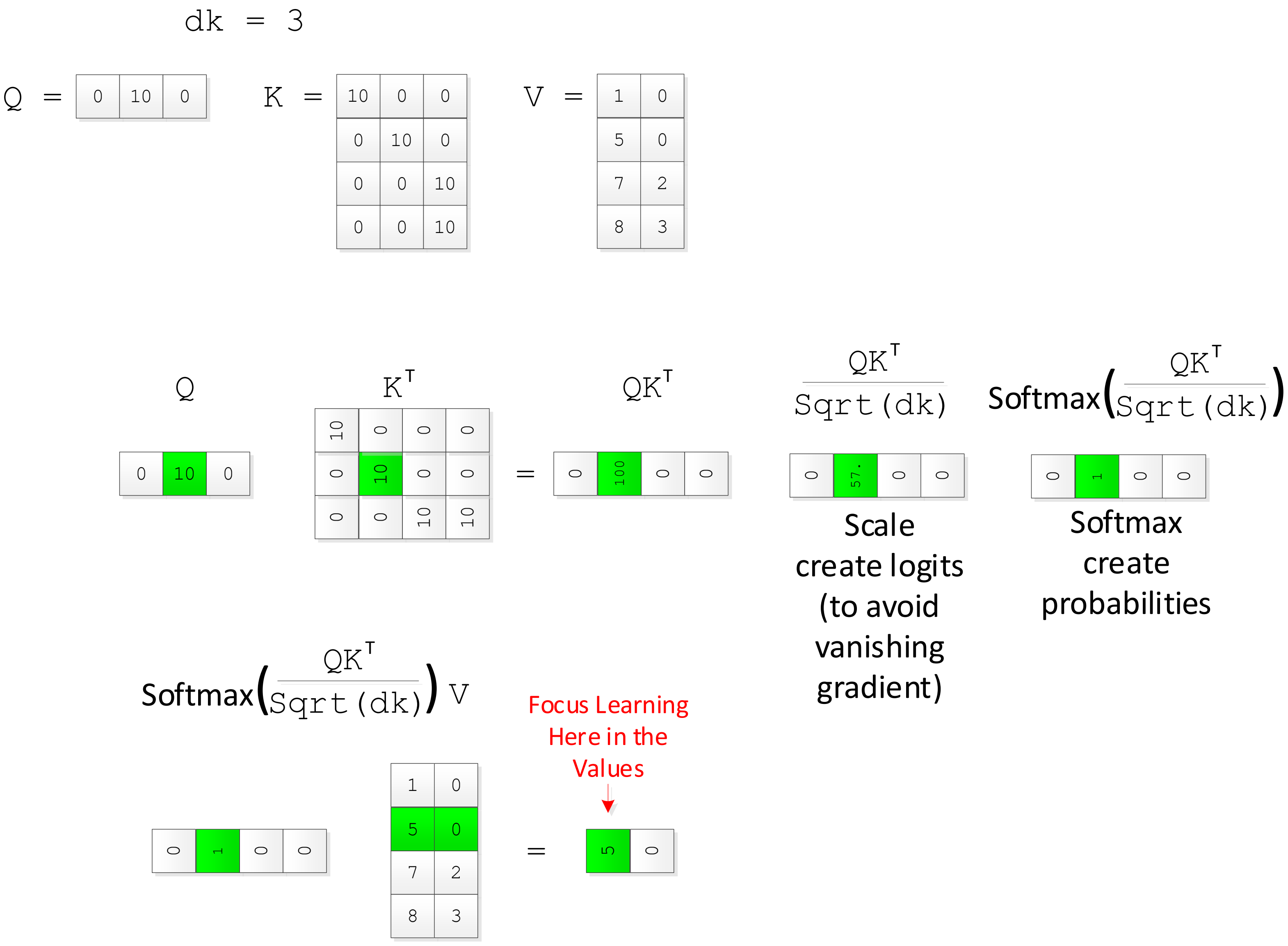
$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{Q\,K^T}{\text{Sqrt}(dk)}\right)\,V$$

How does Attention Work?

The Q, K and V are PyTorch **Linear** layers with bias and learnable parameters that allow for learning the most important values (see slide 16).

Linear – PyTorch 1.12 documentation

(example derived from Wood reference)



$dk = 3$

$$Q = \begin{bmatrix} 0 & 10 & 0 \end{bmatrix} \quad K = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \\ 0 & 0 & 10 \end{bmatrix} \quad V = \begin{bmatrix} 1 & 0 \\ 5 & 0 \\ 7 & 2 \\ 8 & 3 \end{bmatrix}$$

$$\underset{Q}{\begin{bmatrix} 0 & 10 & 0 \end{bmatrix}} \; \underset{K^T}{\begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 10 \end{bmatrix}} = \underset{QK^T}{\begin{bmatrix} 0 & 100 & 0 & 0 \end{bmatrix}}$$

$$\frac{QK^T}{Sqrt(dk)}: \begin{bmatrix} 0 & 57. & 0 & 0 \end{bmatrix}$$

Scale create logits (to avoid vanishing gradient)

$$Softmax\left(\frac{QK^T}{Sqrt(dk)}\right): \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$$

Softmax create probabilities

$$Softmax\left(\frac{QK^T}{Sqrt(dk)}\right) V$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 5 & 0 \\ 7 & 2 \\ 8 & 3 \end{bmatrix} = \begin{bmatrix} 5 & 0 \end{bmatrix}$$

Focus Learning Here in the Values

Apply attention to **focus** on important values

minGPT – Data Input (NLP)

“Before we proceed any further, hear me speak.”

Chunk of Text (block_size + 1)

X	B	e	f	o	r	e		w	e		p	r	
Y		e	f	o	r	e		w	e		p	r	o

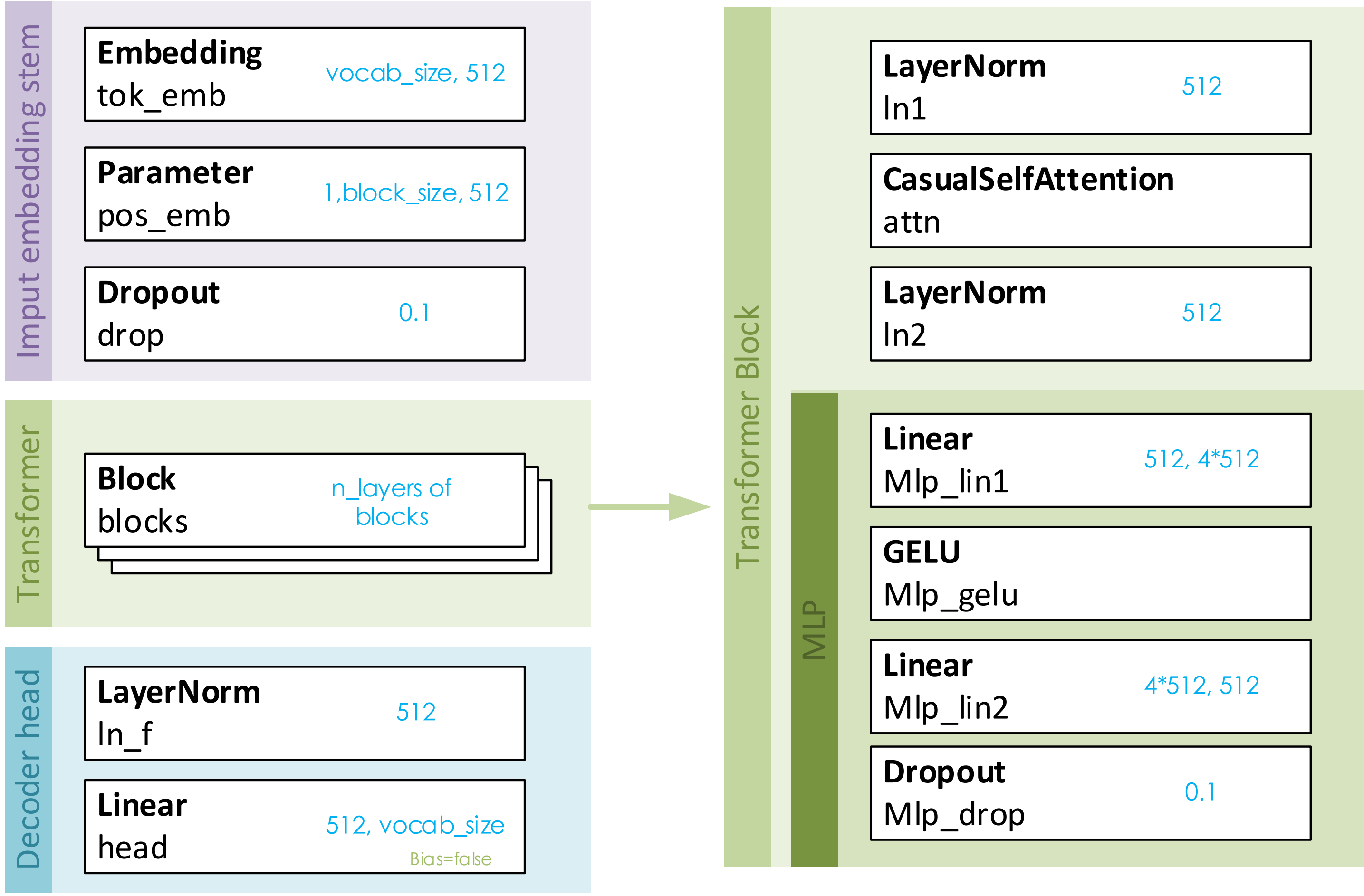
↓

Chunk of ASCII (block_size + 1)

X	66	101	102	111	114	101	32	119	101	32	112	114	
Y		101	102	111	114	101	32	119	101	32	112	114	111

X = training input
Y = training target

minGPT – General Model



Imput embedding stem

Embedding vocab_size, 512
tok_emb

Parameter 1,block_size, 512
pos_emb

Dropout 0.1
drop

Transformer

Block n_layers of blocks
blocks

Decoder head

LayerNorm 512
ln_f

Linear 512, vocab_size
head
Bias=false

Transformer Block

LayerNorm 512
ln1

CasualSelfAttention
attn

LayerNorm 512
ln2

MLP

Linear 512, 4*512
Mlp_lin1

GELU
Mlp_gelu

Linear 4*512, 512
Mlp_lin2

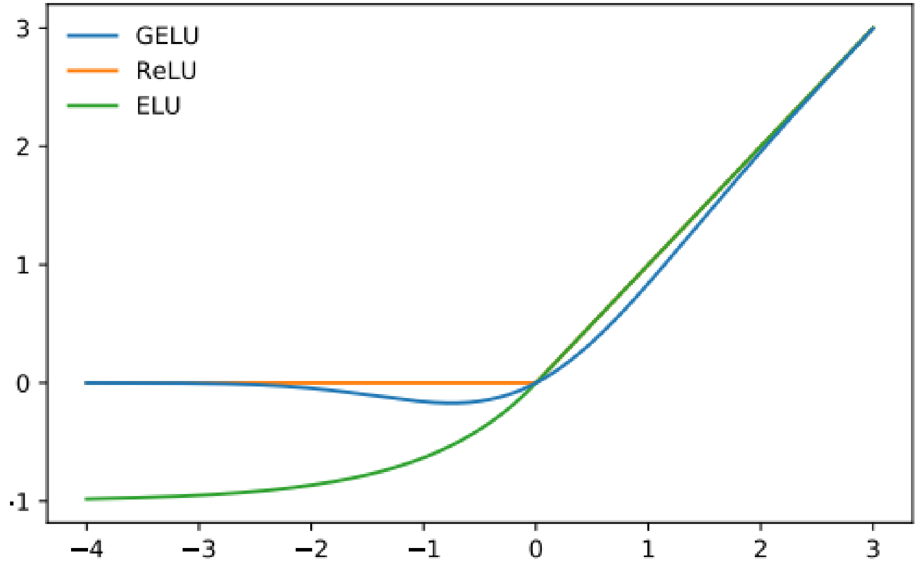
Dropout 0.1
Mlp_drop

minGPT – General Model

GELU (Gaussian Error Linear Unit) is an activation function.

(see Goel reference)

$$\begin{aligned}\mathrm{GELU}(x) &= xP(X \leq x) = x\Phi(x)\\ &\approx 0.5x\left(1 + \tanh\left[\sqrt{2/\pi}\left(x + 0.044715x^3\right)\right]\right)\end{aligned}$$



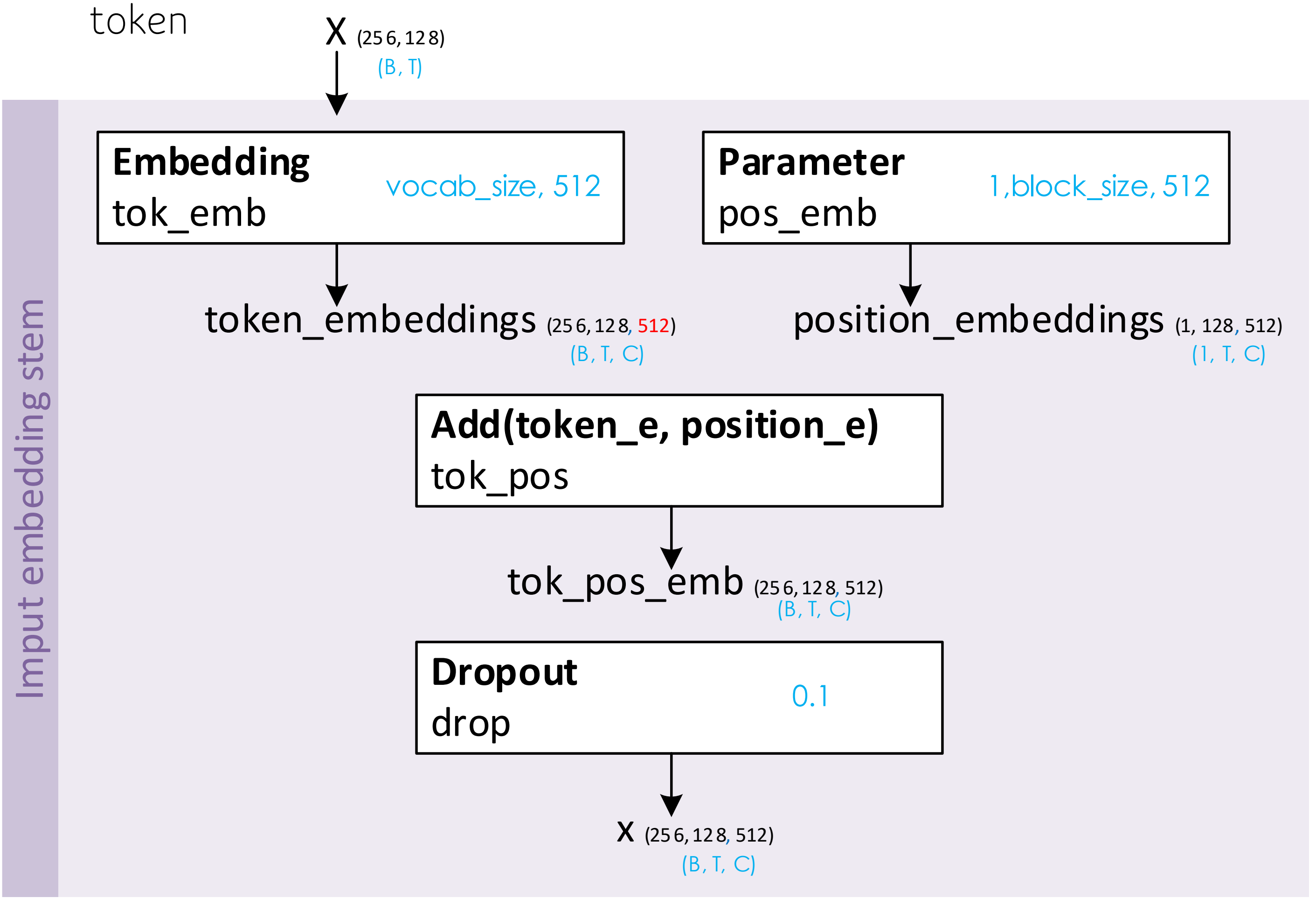
GELU (μ=0, σ=1), ReLU and ELU (α=1)

minGPT – Input Embedding Stem

B = batch size
T = sequence length
C = embed dim

Token embedding (tok_emb) learns embedding of each token

Positional encoding (pos_emb) learns token positioning



Imput embedding stem

X (256, 128)
(B, T)

Embedding vocab_size, 512
tok_emb

Parameter 1,block_size, 512
pos_emb

token_embeddings (256, 128, 512)
(B, T, C)

position_embeddings (1, 128, 512)
(1, T, C)

Add(token_e, position_e)
tok_pos

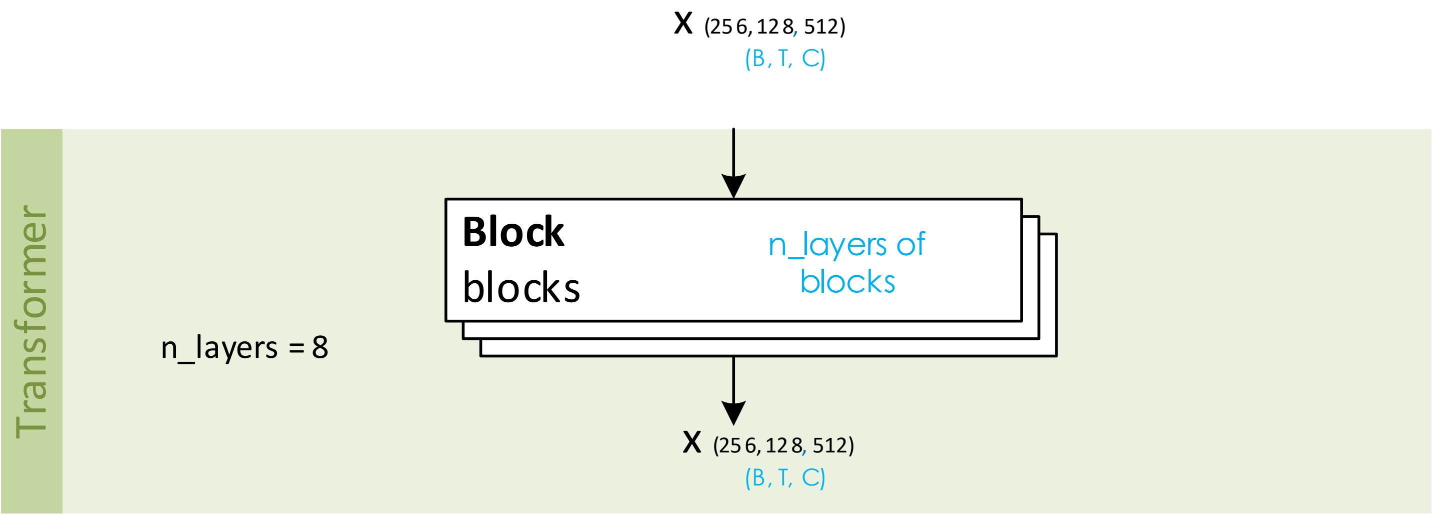
tok_pos_emb (256, 128, 512)
(B, T, C)

Dropout 0.1
drop

X (256, 128, 512)
(B, T, C)

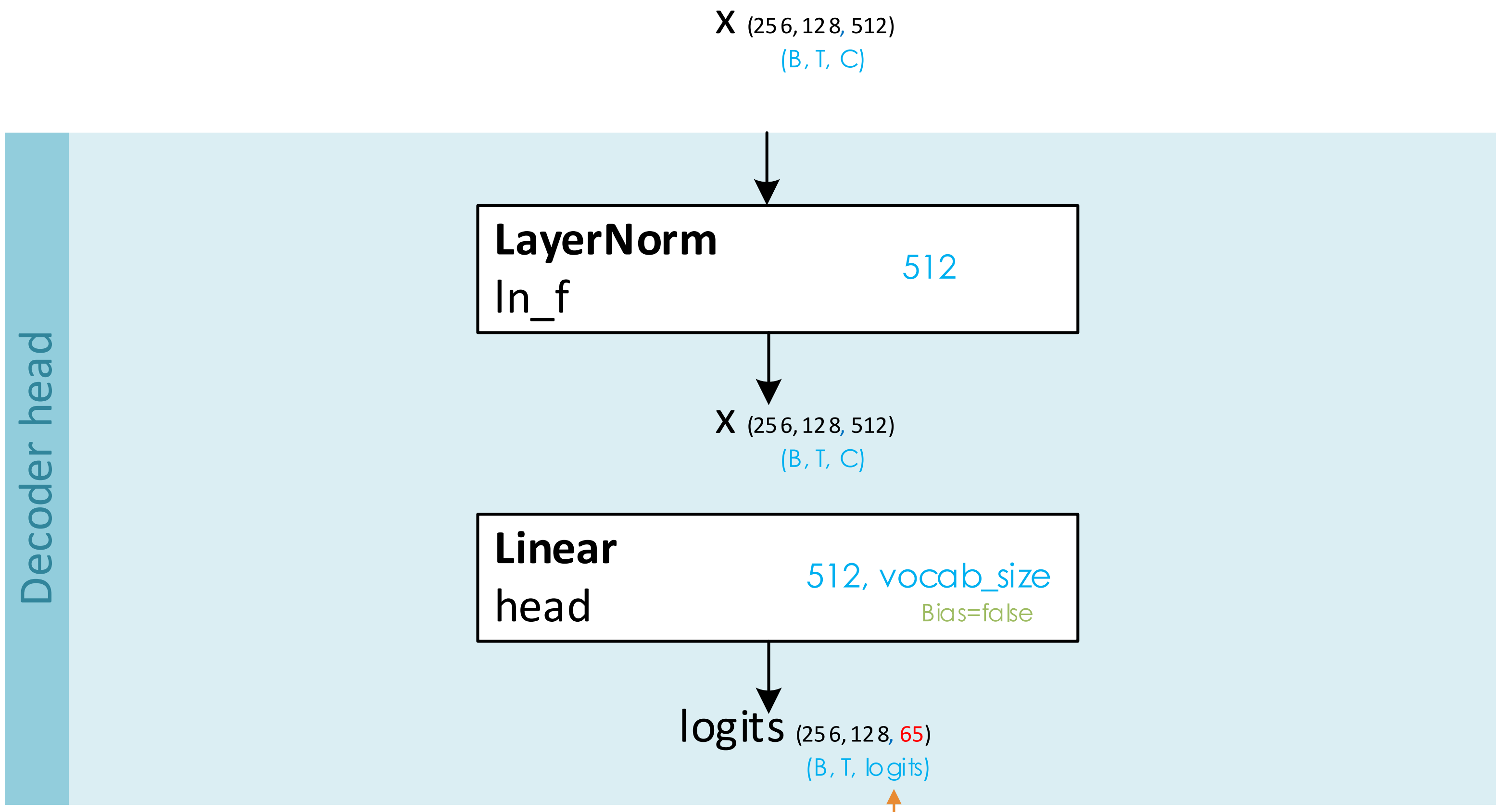
Each character in a sentence is a token, other models (like BERT) use WordPiece tokenization.

minGPT - Transformer



B = batch size
T = sequence length
C = embed dim

minGPT - Decoder Head



X (256, 128, 512)
(B, T, C)

Decoder head

LayerNorm
ln_f
512

X (256, 128, 512)
(B, T, C)

Linear
head
512, vocab_size
Bias=false

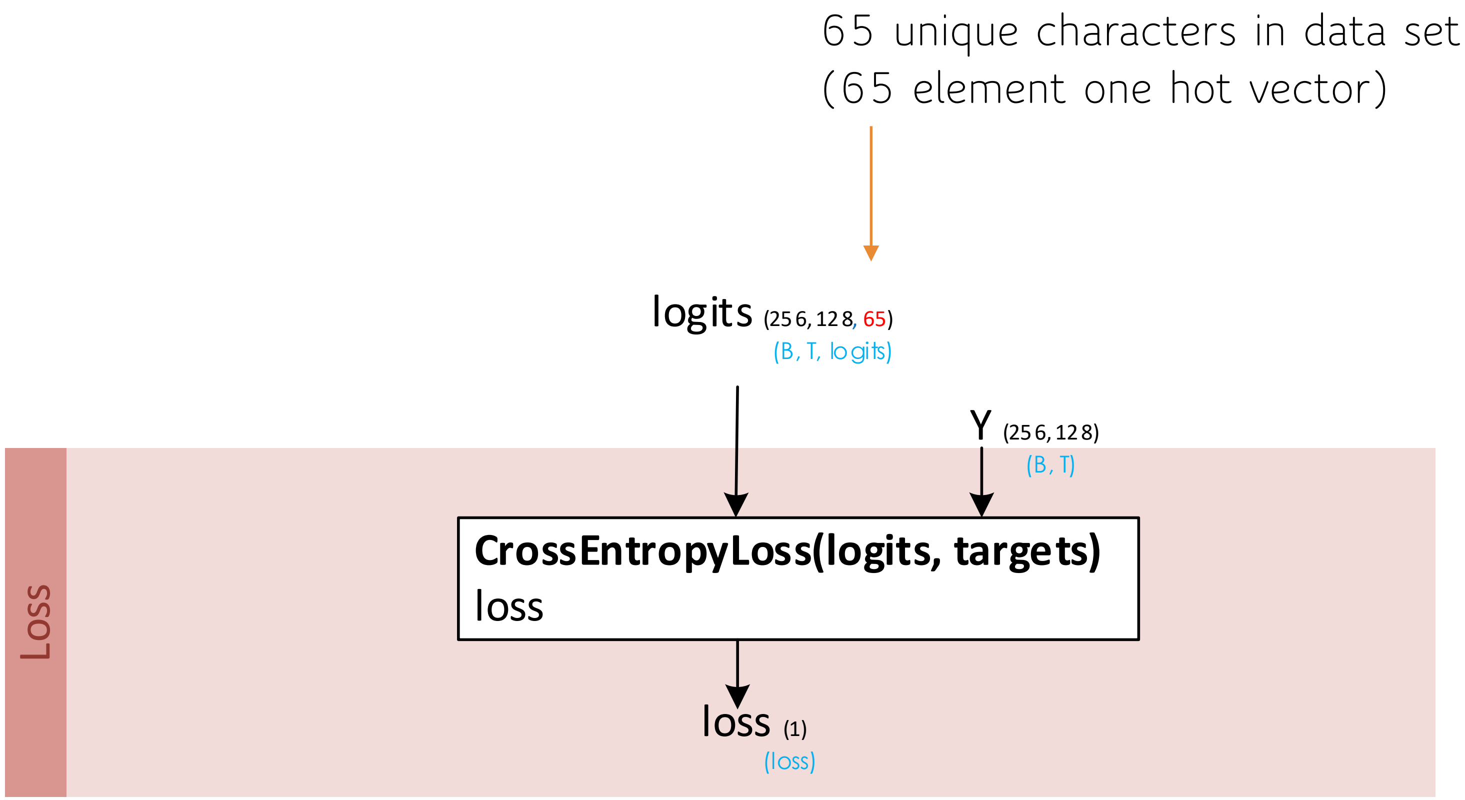
logits (256, 128, 65)
(B, T, logits)

65 unique characters in data set
(65 element one hot vector)

B = batch size
T = sequence length
C = embed dim

minGPT - Loss

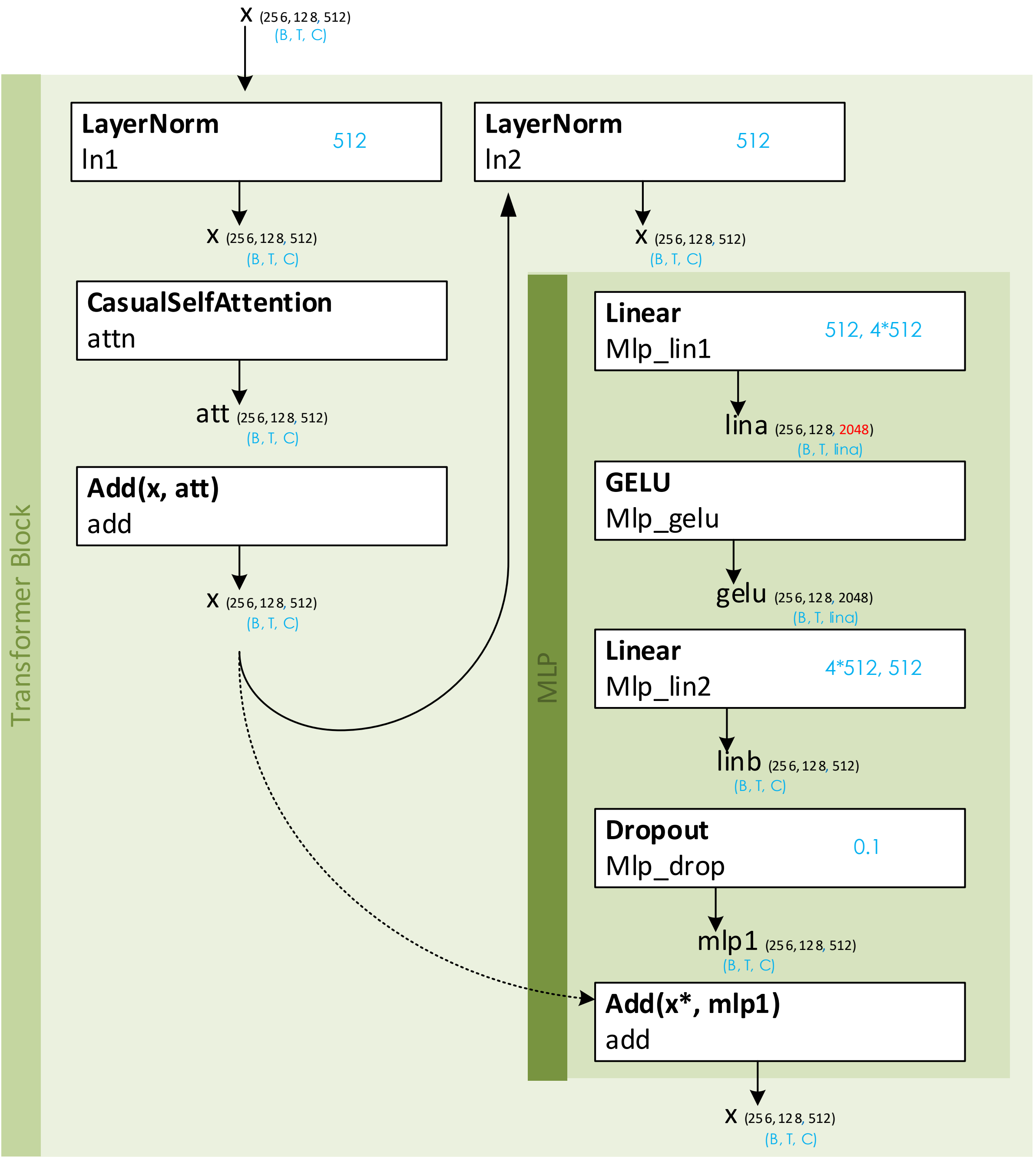
B = batch size
T = sequence length
Logit = one-hot vector


65 unique characters in data set
(65 element one hot vector)
logits (256, 128, 65)
(B, T, logits)
Y (256, 128)
(B, T)
Loss
CrossEntropyLoss(logits, targets)
loss
loss (1)
(loss)

minGPT - Transformer Block

8 blocks are used in character model for Shakespeare Sonnet

B = batch size
T = sequence length
C = embed dim



x (256, 128, 512)
(B, T, C)

Transformer Block

LayerNorm
ln1
512

x (256, 128, 512)
(B, T, C)

CasualSelfAttention
attn

att (256, 128, 512)
(B, T, C)

Add(x, att)
add

x (256, 128, 512)
(B, T, C)

LayerNorm
ln2
512

x (256, 128, 512)
(B, T, C)

MLP

Linear
Mlp_lin1
512, 4*512

lina (256, 128, 2048)
(B, T, lina)

GELU
Mlp_gelu

gelu (256, 128, 2048)
(B, T, lina)

Linear
Mlp_lin2
4*512, 512

linb (256, 128, 512)
(B, T, C)

Dropout
Mlp_drop
0.1

mlp1 (256, 128, 512)
(B, T, C)

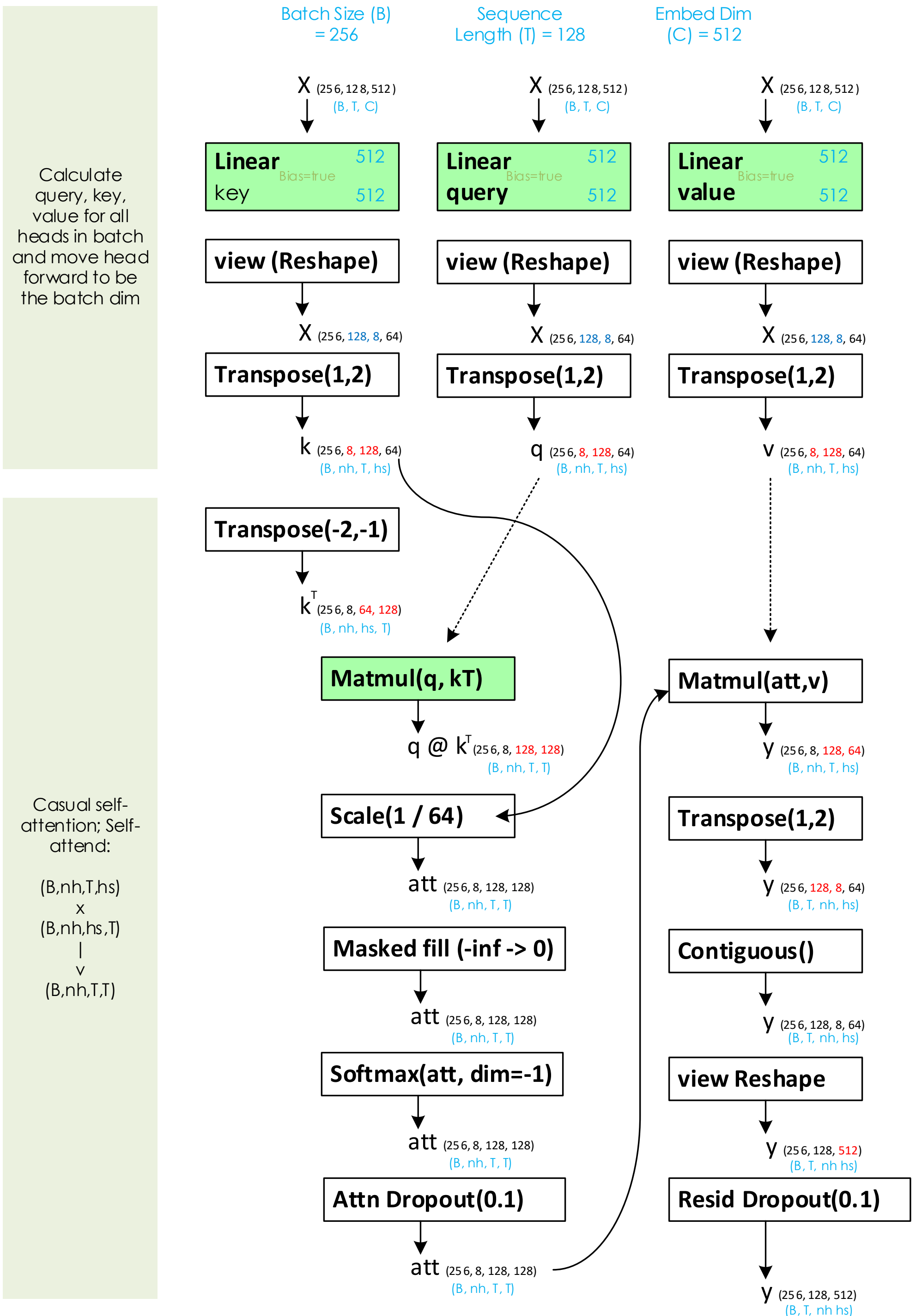
Add(x*, mlp1)
add

x (256, 128, 512)
(B, T, C)

minGPT - Casual Self Attention

Matmul(q, kT) focuses the attention…

…on the important aspects 'learned' in the Linear layers for Key, Query, and Value.



Calculate query, key, value for all heads in batch and move head forward to be the batch dim

Casual self-attention; Self-attend:

(B,nh,T,hs)
x
(B,nh,hs,T)
|
v
(B,nh,T,T)

Batch Size (B) = 256

Sequence Length (T) = 128

Embed Dim (C) = 512

X (256,128,512) (B, T, C)
Linear key 512 Bias=true 512
view (Reshape)
X (256, 128, 8, 64)
Transpose(1,2)
k (256, 8, 128, 64) (B, nh, T, hs)
Transpose(-2,-1)
k^T (256, 8, 64, 128) (B, nh, hs, T)

X (256,128,512) (B, T, C)
Linear query 512 Bias=true 512
view (Reshape)
X (256, 128, 8, 64)
Transpose(1,2)
q (256, 8, 128, 64) (B, nh, T, hs)

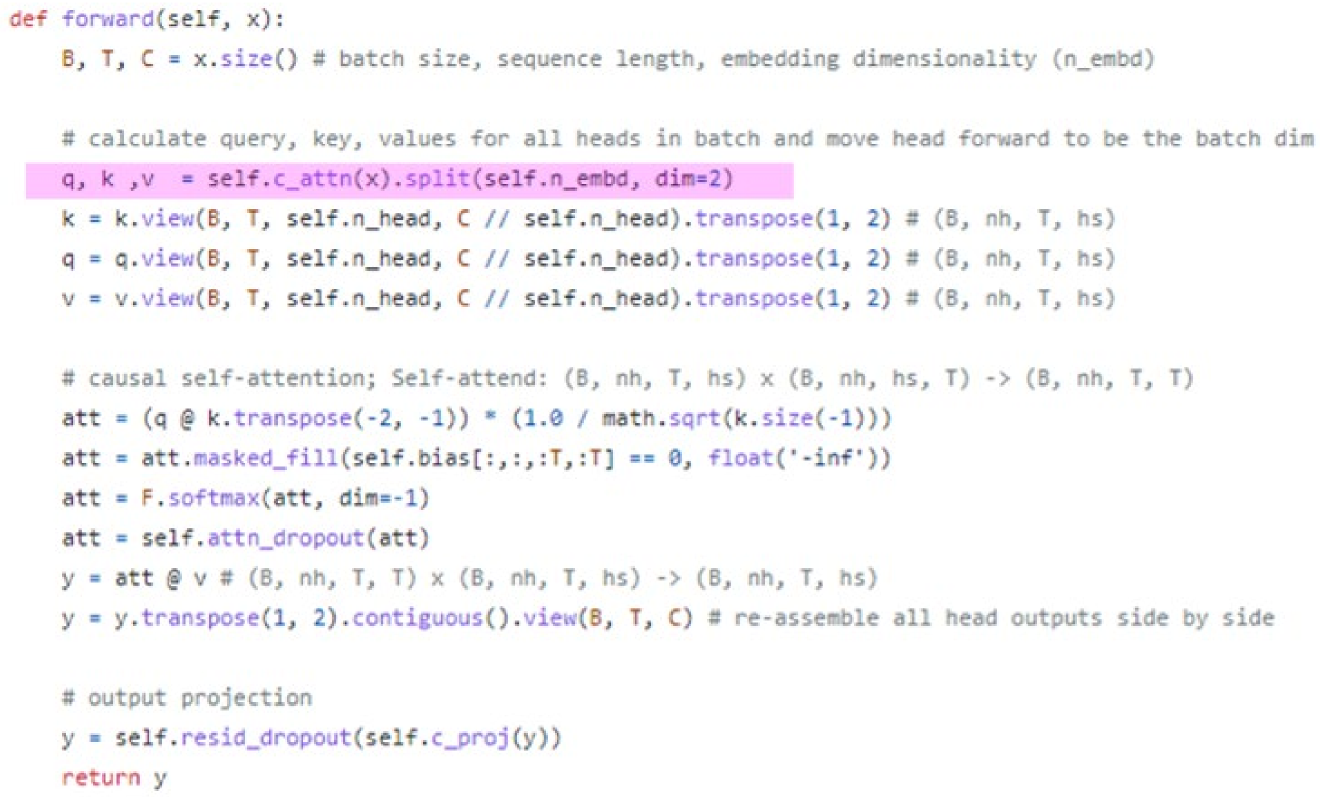
X (256,128,512) (B, T, C)
Linear value 512 Bias=true 512
view (Reshape)
X (256, 128, 8, 64)
Transpose(1,2)
v (256, 8, 128, 64) (B, nh, T, hs)

Matmul(q, kT)
q @ k^T (256, 8, 128, 128) (B, nh, T, T)
Scale(1 / 64)
att (256, 8, 128, 128) (B, nh, T, T)
Masked fill (-inf -> 0)
att (256, 8, 128, 128) (B, nh, T, T)
Softmax(att, dim=-1)
att (256, 8, 128, 128) (B, nh, T, T)
Attn Dropout(0.1)
att (256, 8, 128, 128) (B, nh, T, T)

Matmul(att,v)
y (256, 8, 128, 64) (B, nh, T, hs)
Transpose(1,2)
y (256, 128, 8, 64) (B, T, nh, hs)
Contiguous()
y (256, 128, 8, 64) (B, T, nh, hs)
view Reshape
y (256, 128, 512) (B, T, nh hs)
Resid Dropout(0.1)
y (256, 128, 512) (B, T, nh hs)

minGPT - Transformer Block

CasualSelfAttention forward pass
(from minGPT github)



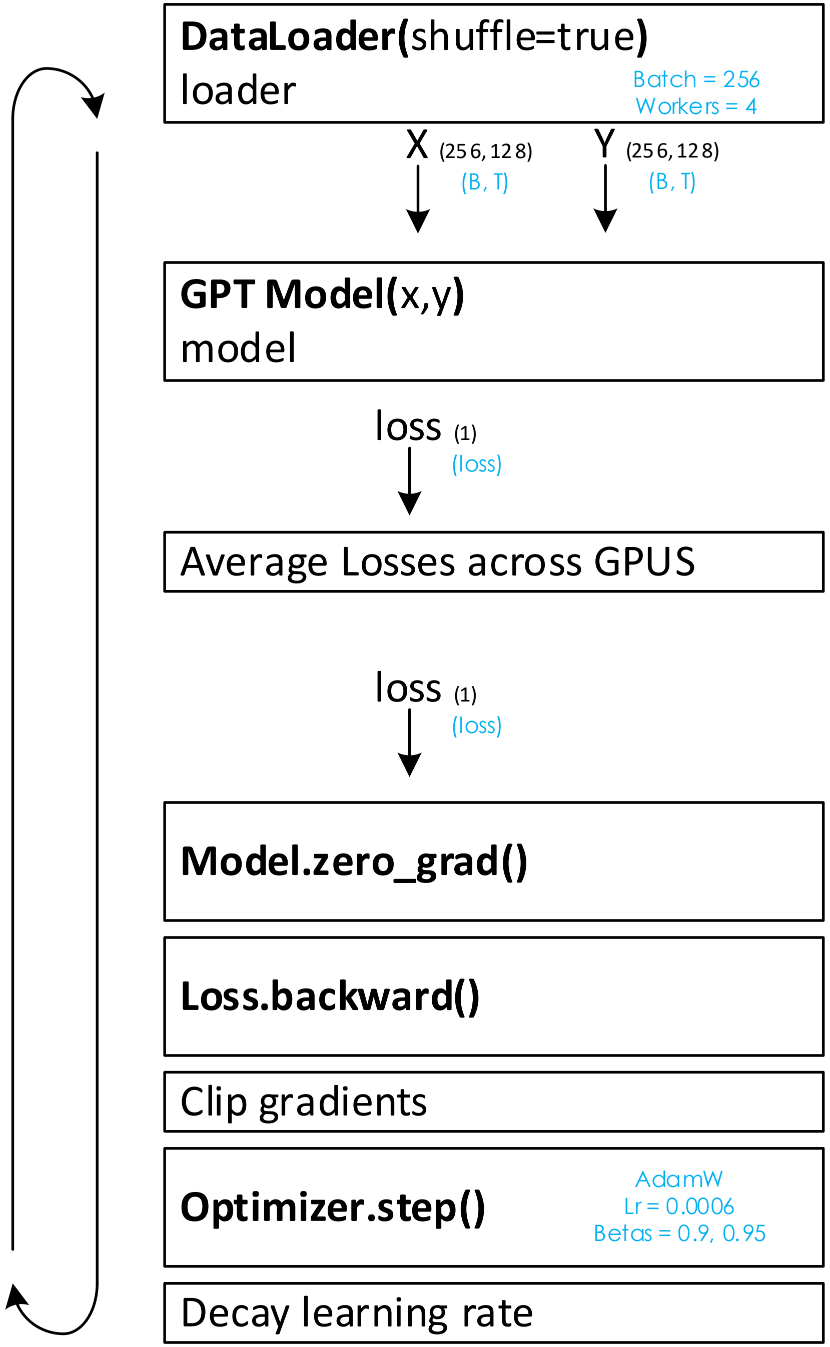
```
def forward(self, x):
    B, T, C = x.size() # batch size, sequence length, embedding dimensionality (n_embd)

    # calculate query, key, values for all heads in batch and move head forward to be the batch dim
    q, k ,v  = self.c_attn(x).split(self.n_embd, dim=2)
    k = k.view(B, T, self.n_head, C // self.n_head).transpose(1, 2) # (B, nh, T, hs)
    q = q.view(B, T, self.n_head, C // self.n_head).transpose(1, 2) # (B, nh, T, hs)
    v = v.view(B, T, self.n_head, C // self.n_head).transpose(1, 2) # (B, nh, T, hs)

    # causal self-attention; Self-attend: (B, nh, T, hs) x (B, nh, hs, T) -> (B, nh, T, T)
    att = (q @ k.transpose(-2, -1)) * (1.0 / math.sqrt(k.size(-1)))
    att = att.masked_fill(self.bias[:,:,:T,:T] == 0, float('-inf'))
    att = F.softmax(att, dim=-1)
    att = self.attn_dropout(att)
    y = att @ v # (B, nh, T, T) x (B, nh, T, hs) -> (B, nh, T, hs)
    y = y.transpose(1, 2).contiguous().view(B, T, C) # re-assemble all head outputs side by side

    # output projection
    y = self.resid_dropout(self.c_proj(y))
    return y
```

B = batch size
T = sequence length
C = embed dim

minGPT - Training

Weights **do not decay** on LayerNorm, Embedding and Position Embedding layers.

B = batch size
T = sequence length
C = embed dim



DataLoader(shuffle=true**)**
loader
Batch = 256
Workers = 4

X (256,128)
(B, T)

Y (256,128)
(B, T)

GPT Model(x,y**)**
model

loss (1)
(loss)

Average Losses across GPUS

loss (1)
(loss)

Model.zero_grad()

Loss.backward()

Clip gradients

Optimizer.step()
AdamW
Lr = 0.0006
Betas = 0.9, 0.95

Decay learning rate

Single Cycle within epoch

References

minGPT; Andrej Karpathy; GitHub karpathy/minGPT; 2020; https://github.com/karpathy/minGPT

GPT-2; OpenAI; GitHub openai/gpt-2; 2019; https://github.com/openai/gpt-2

Language Models are Unsupervised Multitask Learners; Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever; 2019; https://paperswithcode.com/paper/language-models-are-unsupervised-multitask

Language Models are Few-Shot Learners; Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei; 2020; https://arxiv.org/abs/2005.14165

Image GPT; OpenAI; "Image GPT"; 2020; https://openai.com/blog/image-gpt/

Generative Pretraining from Pixels; Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhiriwal, David Luan, Ilya Sutskever; 2020; https://cdn.openai.com/papers/Generative_Pretraining_from_Pixels_V2.pdf

HuggingFace Site; https://huggingface.co/; tons of great transformer models here!

References (Transformer Models)

What is a Transformer Neural Network?; Thomas Wood; DeepAI; 202x; https://deepai.org/machine-learning-glossary-and-terms/transformer-neural-network

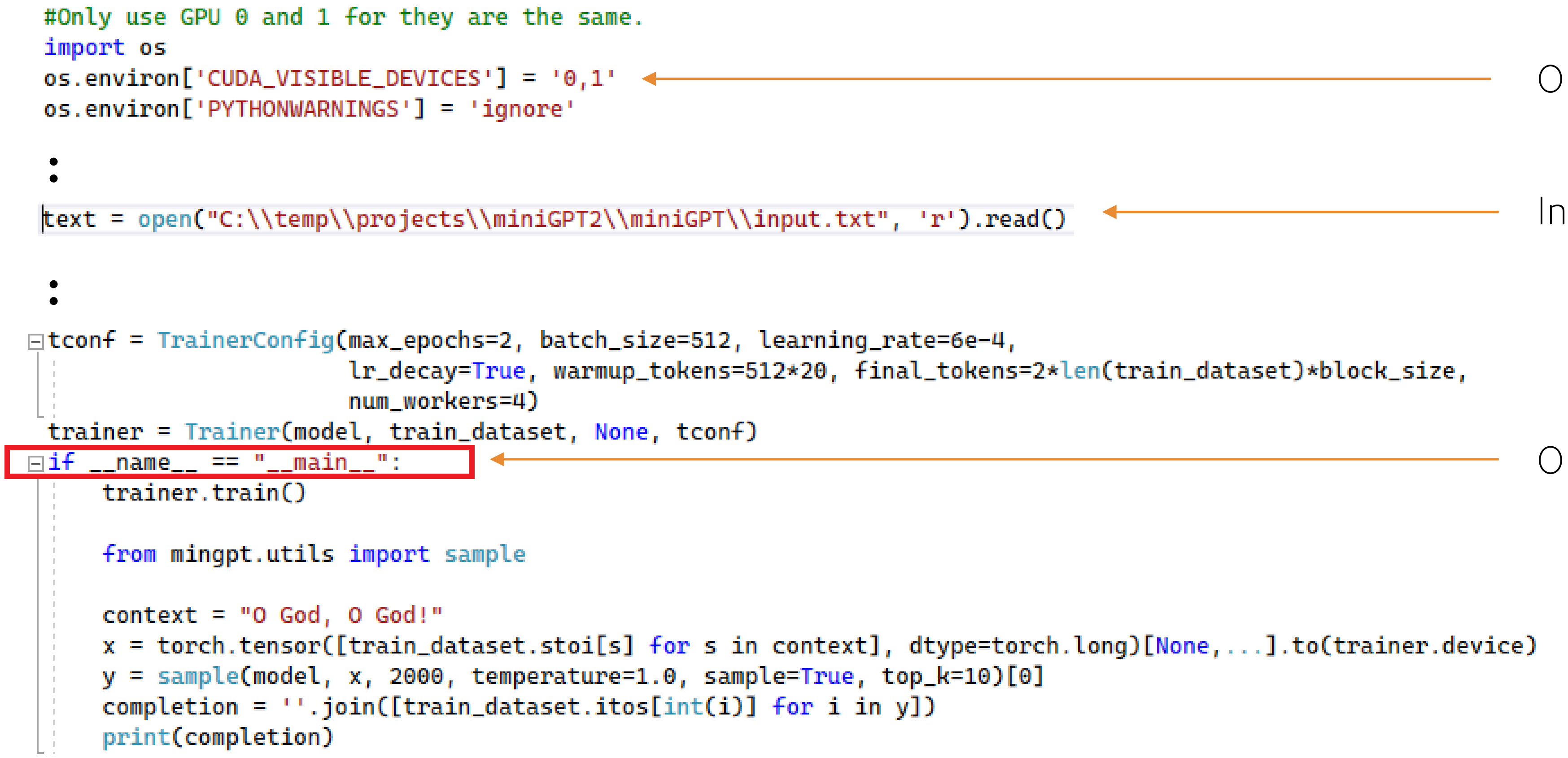
GPT; NVIDIA/FastTransformer; GitHub; 2022; https://github.com/NVIDIA/FasterTransformer/blob/main/docs/gpt_guide.md

References (Attention and GELU)

A Generic Visualization Approach for Convolutional Neural Networks; Ahmed Taha, Xitong Yang, Abhinav Shrivastava, and Larry Davis; https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123620715.pdf

GELU activation; Shaurya Goel; Medium, 2019; https://medium.com/@shauryagoel/gelu-gaussian-error-linear-unit-4ec59fb2e47c

Appendix – Code Modifications for VS2022



```
#Only use GPU 0 and 1 for they are the same.
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '0,1'
os.environ['PYTHONWARNINGS'] = 'ignore'

:

text = open("C:\\temp\\projects\\miniGPT2\\miniGPT\\input.txt", 'r').read()

:

tconf = TrainerConfig(max_epochs=2, batch_size=512, learning_rate=6e-4,
                      lr_decay=True, warmup_tokens=512*20, final_tokens=2*len(train_dataset)*block_size,
                      num_workers=4)
trainer = Trainer(model, train_dataset, None, tconf)
if __name__ == "__main__":
    trainer.train()

    from mingpt.utils import sample

    context = "O God, O God!"
    x = torch.tensor([train_dataset.stoi[s] for s in context], dtype=torch.long)[None,...].to(trainer.device)
    y = sample(model, x, 2000, temperature=1.0, sample=True, top_k=10)[0]
    completion = ''.join([train_dataset.itos[int(i)] for i in y])
    print(completion)
```

Only run on similar GPUs.

Input file location.

Only run main instance.