Mathematic Trick in self-attention.

+ 8 tokens to talk to each other

1: should not future token
(only prev contx to curr time step)

↓ how?

average of proceeding

ie, average M (M = Me)
1st 2nd 3rd

dimension:

B.T.C.

Batch | Time | channel (representation space)

↑ for each token, a vector that represents info

- for each of byn token, calculate average

```
xbow = torch.zeros((B,T,C))
for b in range(B):
    for t in range(T):   <- each token
        xprev = x[b,:t+1]
        xbow[b,t] = torch.mean(xprev, 0)
```

b:th, 0:t+1

0th dimension.

||
EFFICIENCY
-MATH-
||

Implementation, Transformers

Query: question
Key: meaning of myself ↓
(what abt I mean)

+ Query: `nn.Linear(C, head_size, bias=False)`
+ Key: `nn.Linear(C, head_size, bias=False)` / only matrix multiplication

↓ K = key(x)
q = query(x) (feeding input)

Communication bt/ween queries and keys (dot product)

↓

→ wei = q @ k.transpose(-2,-1) (DOT PRODUCT), matrix dimensions first
(B,T,16) @ (B,16,T) → (B,T,T)

↓

1. wei.masked_fill(tril==0, float('-inf'))

[x -inf -inf -inf
  x x -inf -inf
  x x x -inf
  x x x x]

2. wei = F.softmax(wei, dim=-1)

[0.5 0.5 -
  .....]

1. query purpose question.
i.e., looking for vowels.
2. key purpose "answer"
i.e., I am a vowel.
3. when similar,
dot product gives
high interesting score → wei
4. upper-triangular masking to avoid future tokens communicating
5. normalize score i.e., -0.1 ~ w/ softmax
(how much information to aggregate from information of the past)

* but we don't aggregate tokens themselves, but the meaning of them; v
'wei @ v'
softmax'd — meaning (for question) adjusted

Notes:

- There's no idea of space; thus we '+' position_embedding

```
pos_emb = self.position_embedding_table(torch.arange(T, ...))
x = token_emb + pos_emb
```

(B,T,C) (T,C)

- 'tril' (top-triangular mask) is not always the constraint, only for self-regressive Transformers. other e.g., sentiment prediction you don't 'tril', you use an "encoder" block, where no 'tril'
  - we call as "decoder," since we're just decoding language (to next token). you mask for future not talk to present; auto-regressive. & it can be decoder-only or both etc.

- 'self-attention': we use x to generate all 'qkv'
  - 'cross-attention': K,V came from other source & x.
    i.e., encoder block for external ctx
    (to read information from side)
    (different language, different media)

- we use $\sqrt{d_k}$ for $\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$
  - as "scaled attention" to keep softmax diffused & confident. (peaky)

0.19, .14, .21, .14, .28 | .07, .001, .161, .001, .808.

Multi-head Attention:

+ Attend to information from different representation space
  + multiple relationships between tokens
  + multiple heads in parallel

↓

Multi-Layer Perceptrons

+ token-individual level
+ process of updated meaning itself
+ (factual info?)

[nn.Linear(n_embd, n_embd)
nn.ReLU()]

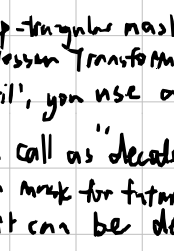
↓

Attention Blocks

+ Attension Heads * n_heads
+ MLP

* Problem: Neural Network too deep *
→ optimization issue

1. Residual Connection



+ A skip connection with addition

(Attention forks off from residual pathway, while addition contributes grad equally.
→ Gradient Highway
(supervision to input)

+ Identity doing nothing)

+ Projection → Linear Transformation (After Head, Feed Forward)
'self.proj = nn.Linear(num_heads * head_size, n_embd)'

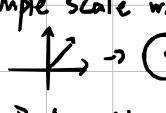
2. Inner-layer of FeedForward should be multiplied by 4

3. LayerNorm: Normalizes Inputs → 0 mean, 1 standard-deviation

→ LayerNorm BEFORE transformation

+ Per-token, w/ trainable parameters

+ conditioning the Addition properties of residual connection
(we're adding things everytime)
+ decouple scale w/ direction (only direction)



1. Before Attention, after Attention

```
def forward(self, x):
    x = x + self.attention_heads(self.layernorm1(x))
    x = x + self.feedforward(self.layernorm2(x))
```

2. After Attention Blocks

4. Dropout!

+ when training, randomly reset some activation of neurons, but disable such behavior on inference
  + no co-adaption (don't depend on certain neurons)
+ scaling, against overfitting

1. after softmax (Head)
2. end of FeedForward (back to residual pathway)
3. end of MultiHead