

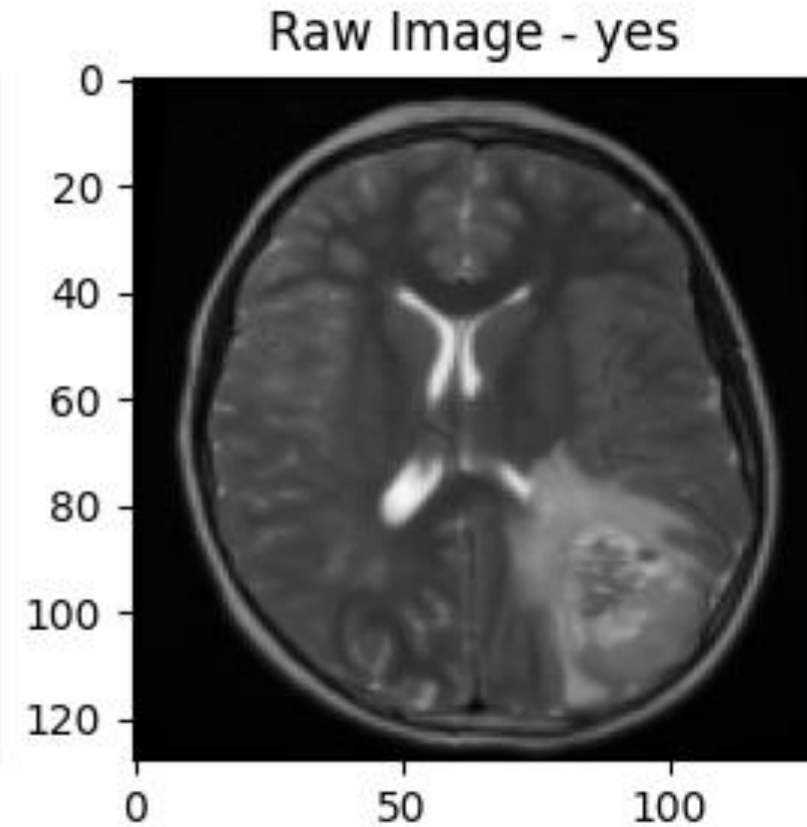
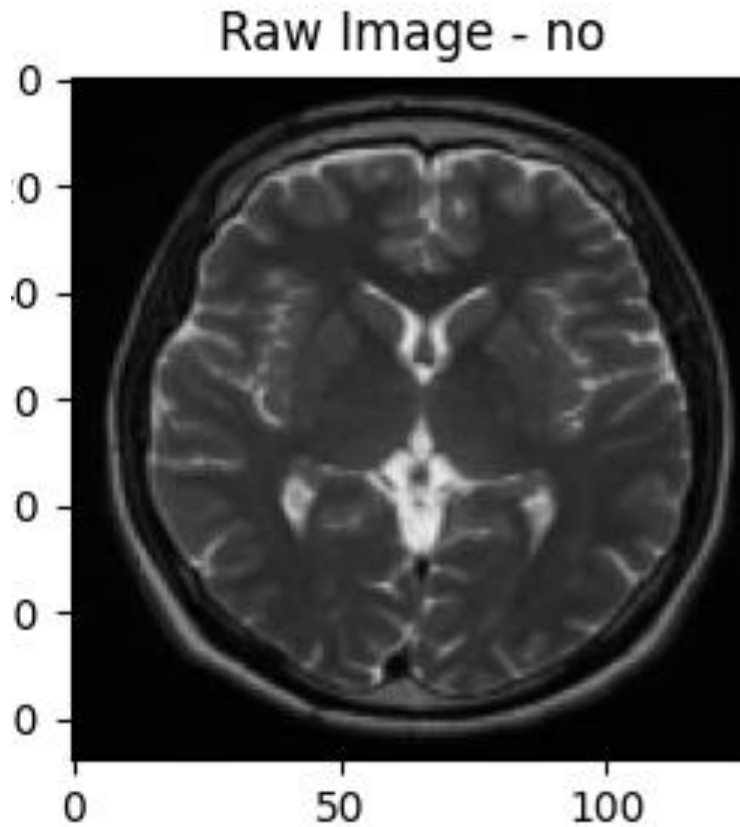


Brain Tumor Detection with Vision Transformers

Ran Minerbi

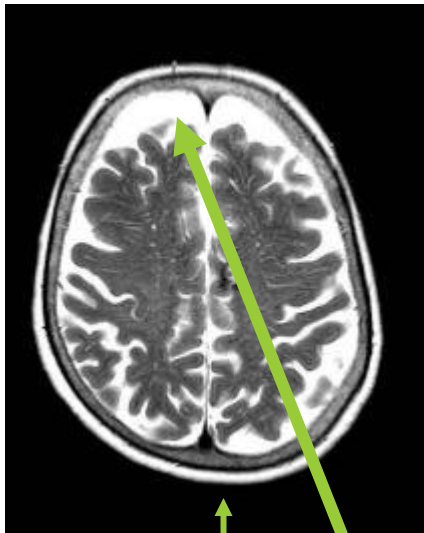
BRAIN TUMOR DETECTION KAGGLE CHALLENGE

Dataset contain 253 MRI
images categorized as
yes/no brain tumor detection
diagnosed



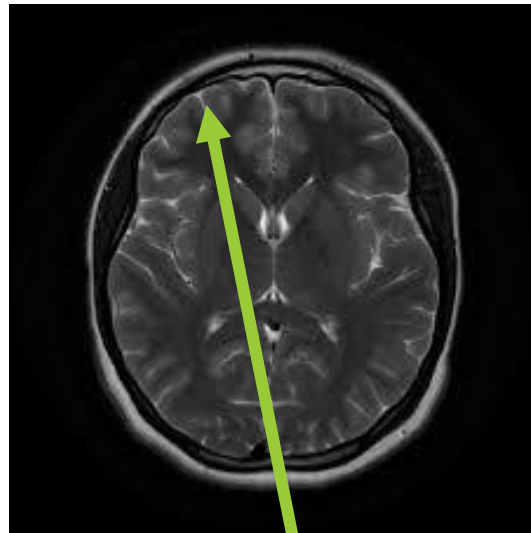
REVIEW DATASET CHALLENGES

- Inconsistent grayscale
- Casing , lobes and skull sometimes black sometimes white

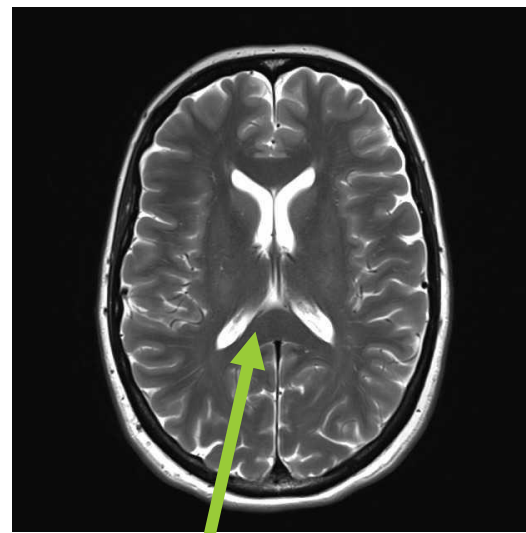


White skull

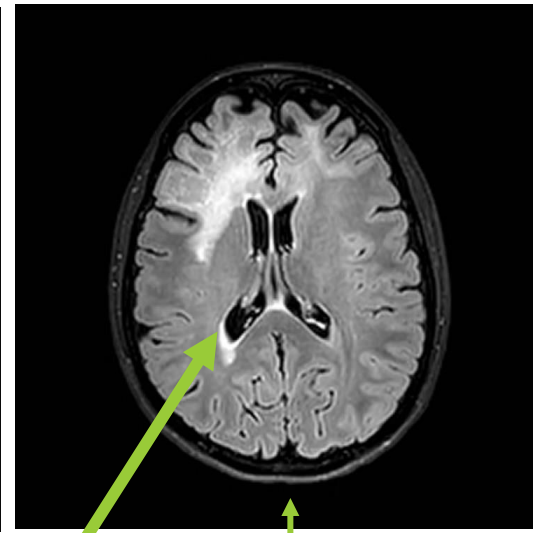
White casing



Black casing



white lobes



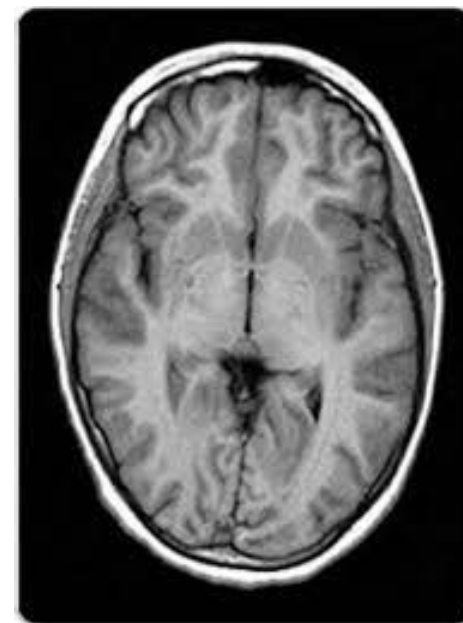
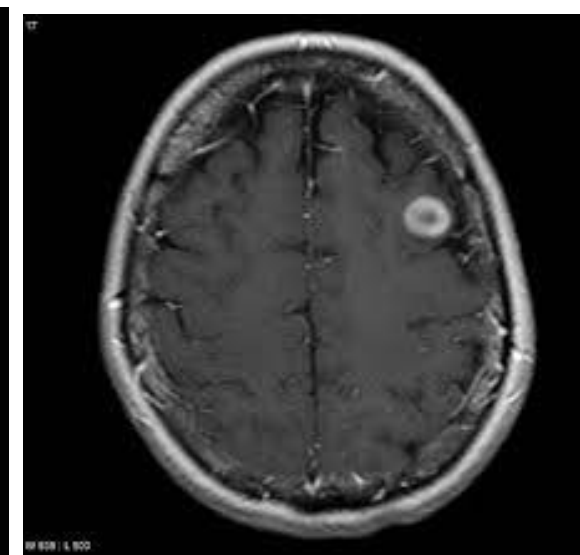
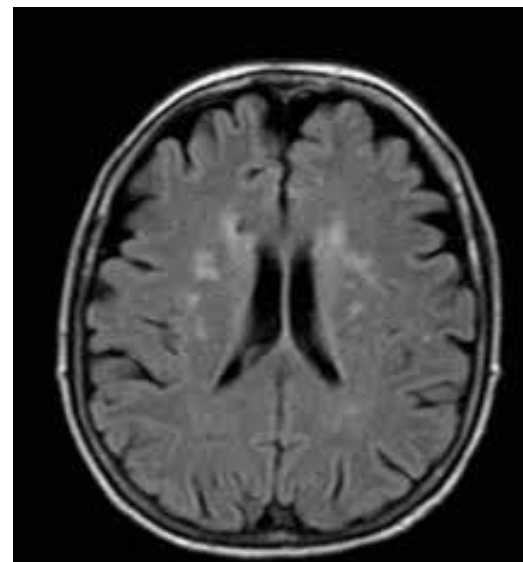
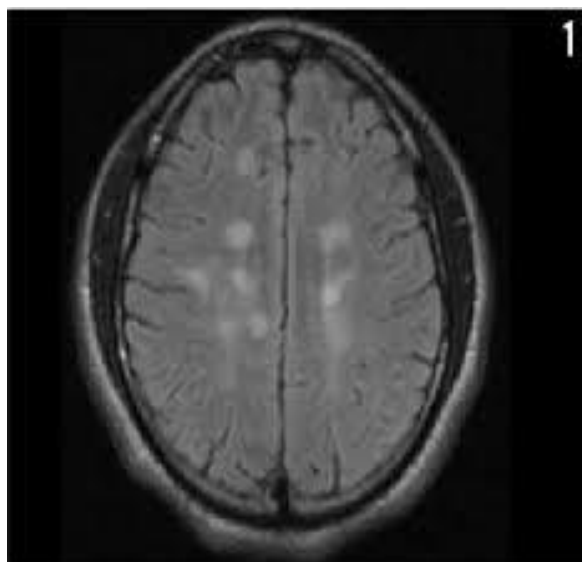
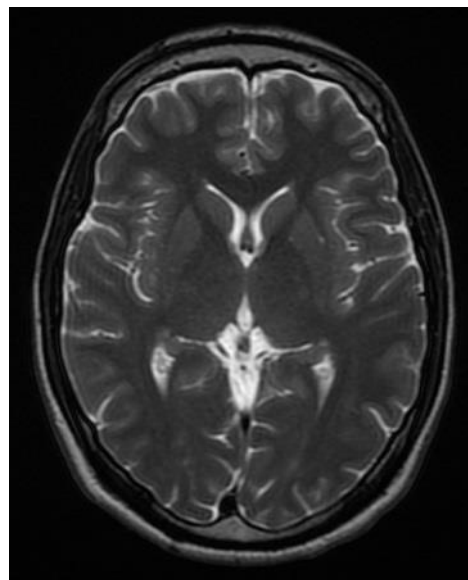
Black lobes

Black skull

REVIEW DATASET CHALLENGES

- Central lobes can be in many shapes or not be at all

No tumor



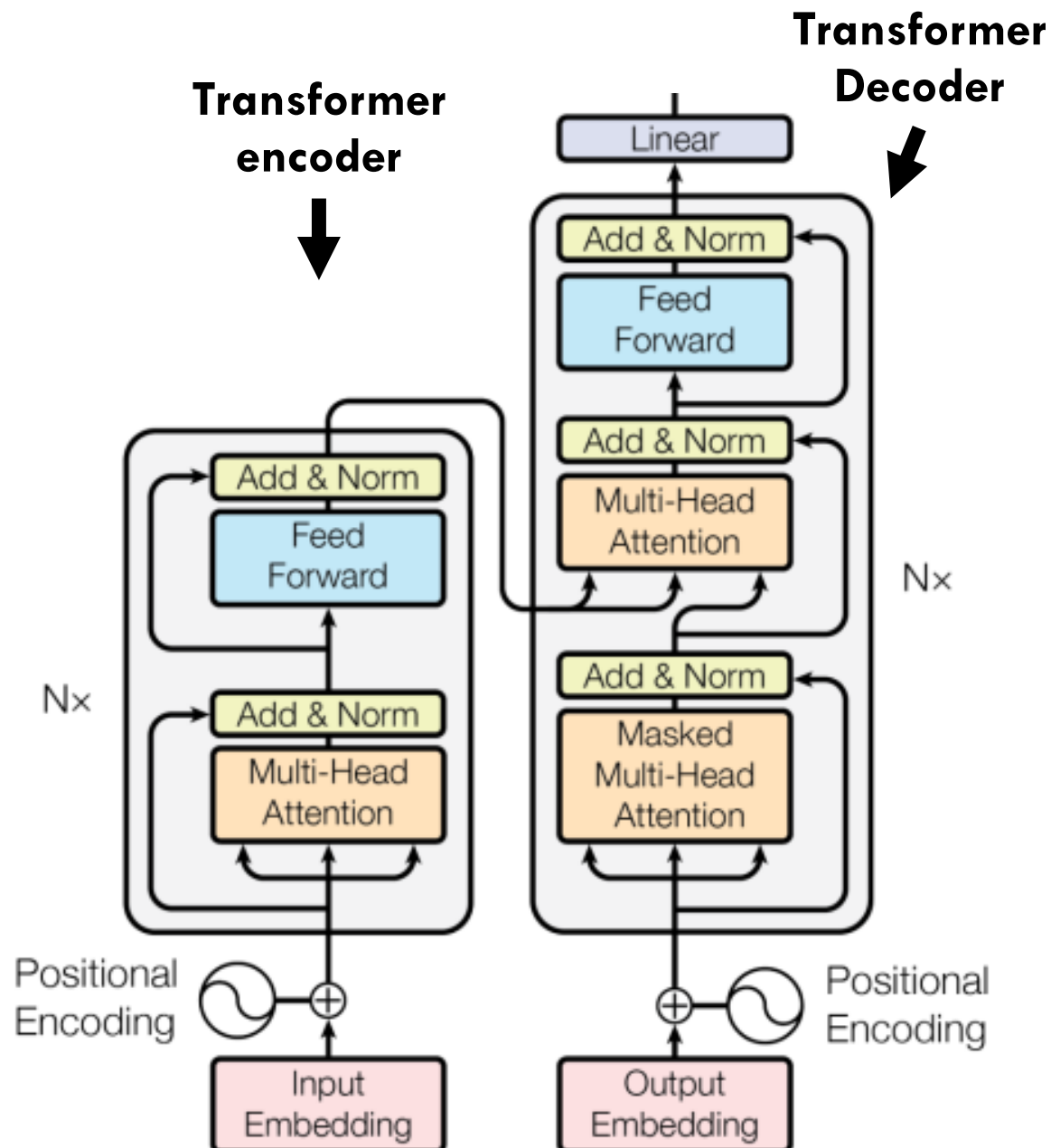
has tumor

TRANSFORMER ARCHITECTURE

Transformer Encoder output issue

Probabilities vector used as input to transformer decoder.

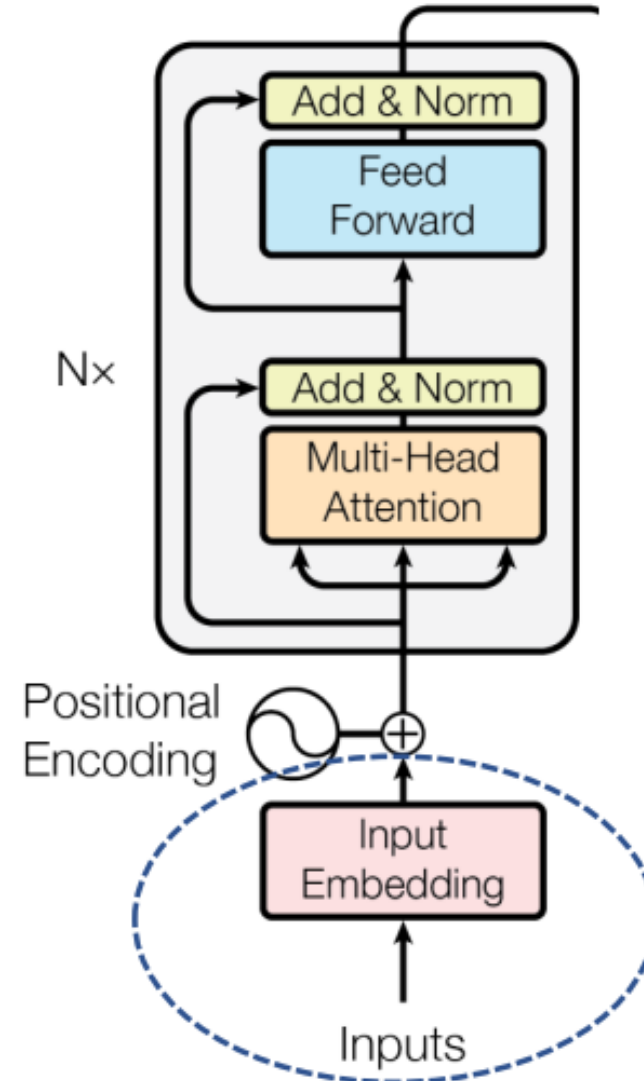
In image classification the encoder is the primary component for classifications



TRANSFORMER ENCODER BLOCK

Transformer encoder block is composed of 3 primary components :

1. **Input embedding**
2. Positional encoding
3. Multi Head Attention
4. output layer



INPUT EMBEDDING LAYER

Input Embedding in NLP

Input embedding block is basically a neural network that is designated to project each

Of the input sequences into a dense numerical representation that the transformer model can process.

For example :

Each word may get 256 vector representation

Whereas “child” – “girl” is equivalent or close to “king”- “queen”

INPUT EMBEDDING IN IMAGES

In vision transformers we split the original image into patches ,

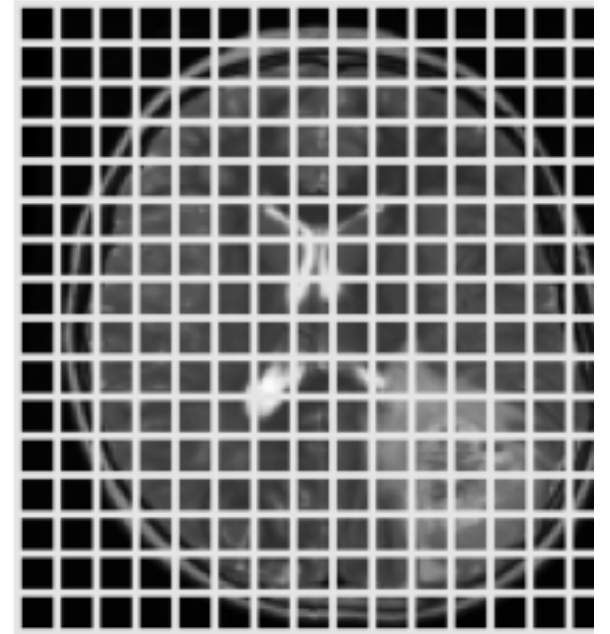
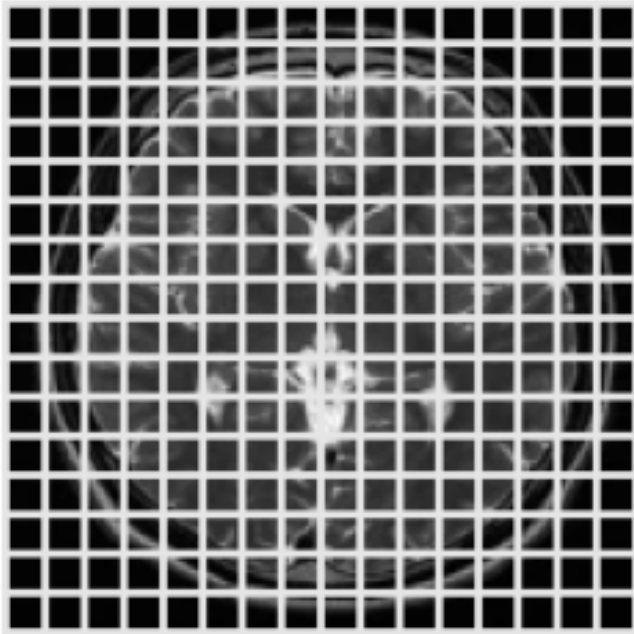
And each patch is projected into vector.

the input embedding more likely to be implemented by CNN rather than fully connected networks.

Each of the image patches is projected through conv 2D network

Where the Output of the 2D Conv network is the patches embedded values

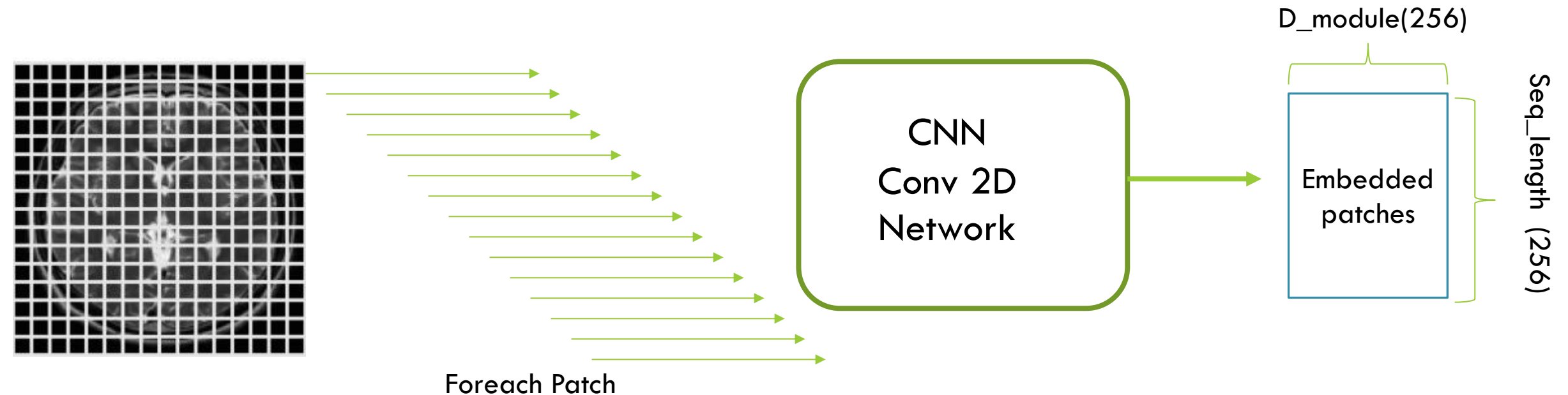
Images as input sequences of patches



STEP 1 - IMAGE PATCHING DIVISION

STEP 2 INPUT EMBEDDING

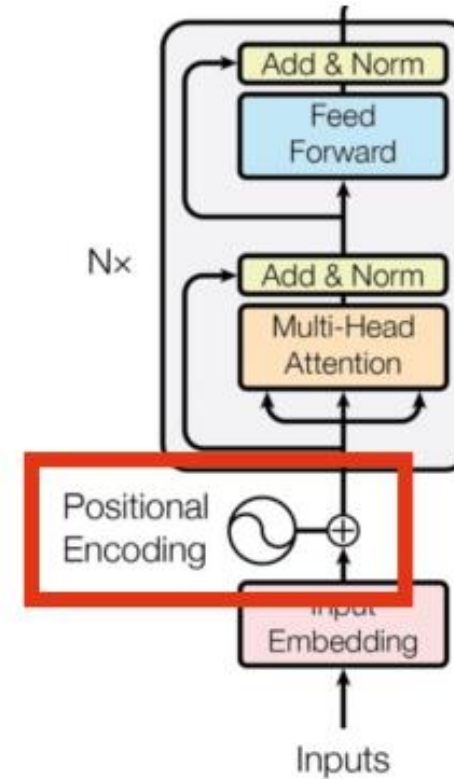
- Each patch from the patched image is projected into $d_module(256)$ sized vector within CNN
- Same content are projected to vase embedded vectors



POSITIONAL ENCODING

Transformer encoder block is composed of 3 primary components :

1. Input embedding
- 2. Positional encoding**
3. Multi Head Attention
4. output layer



POSITIONAL ENCODING CHART FOREACH PATCH

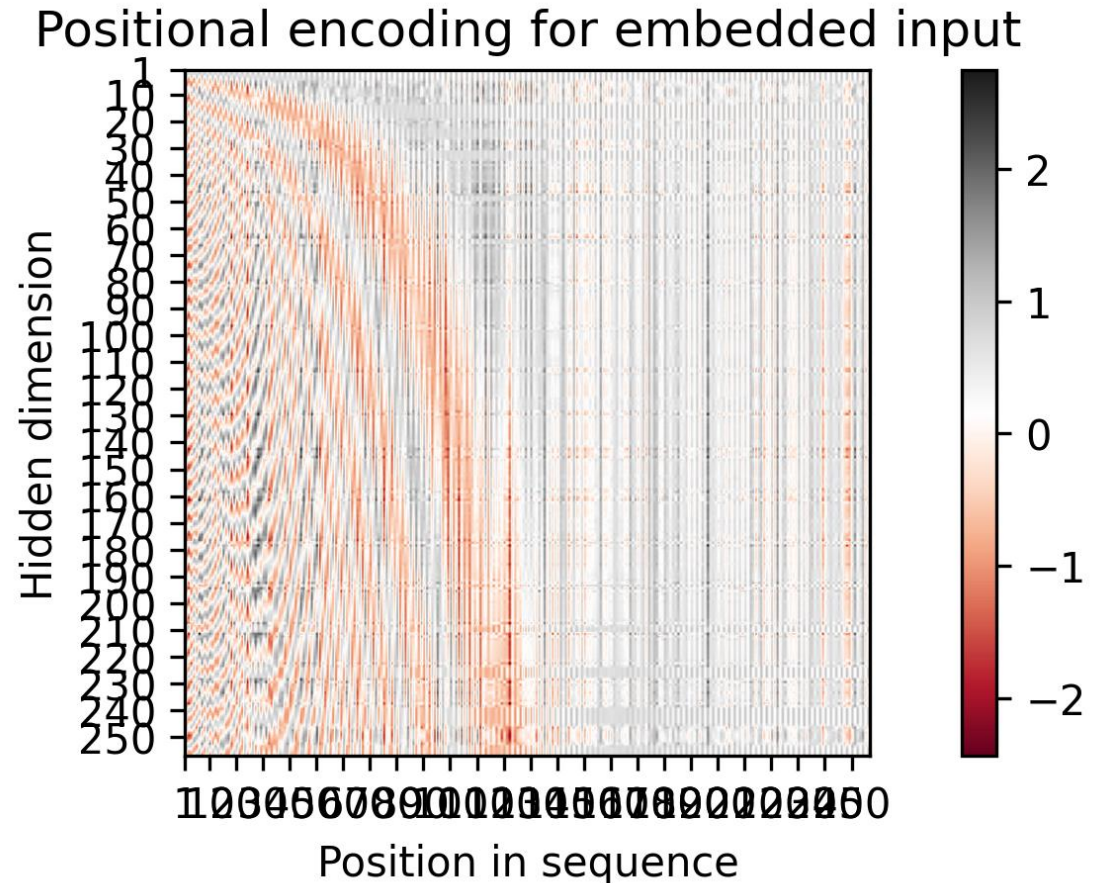
Positional encoding adds to each patch some information regarding its position in the original image.

Helps us to distinct “close” and “far” patches

Helps to represent patterns that can be learned by our model

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

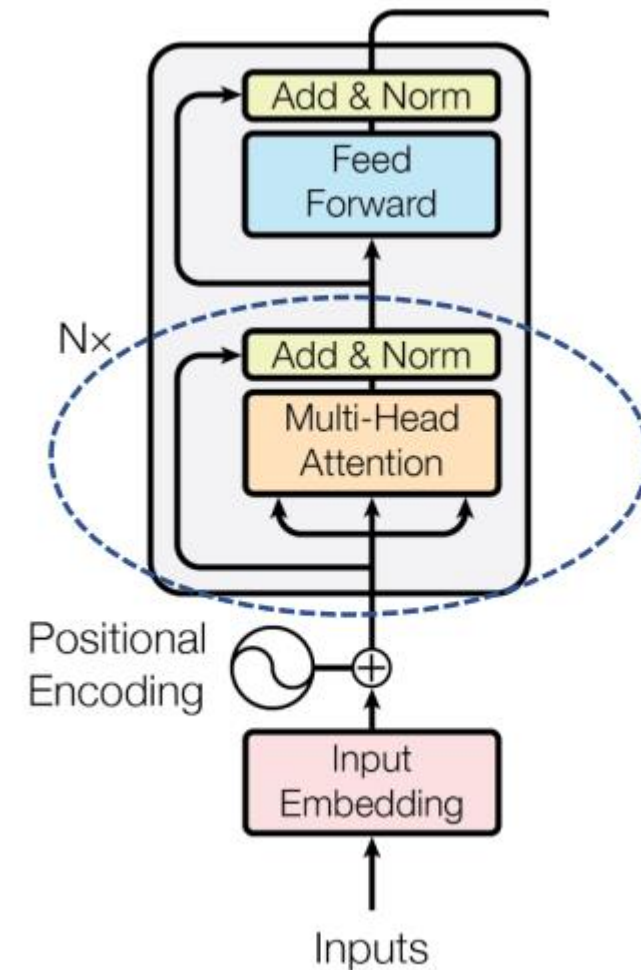
$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$



MULTI HEAD ATTENTION

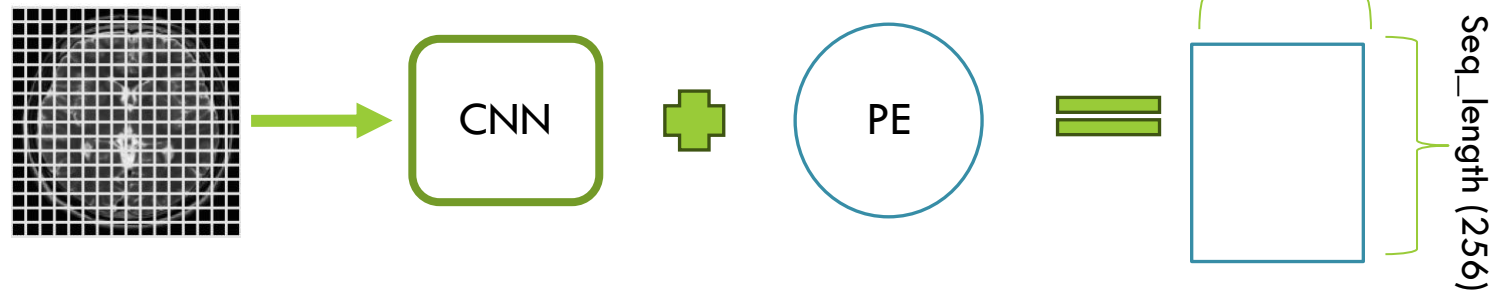
Transformer encoder block is composed of 3 primary components :

1. Input embedding
2. Positional encoding
3. Multi Head Attention
4. output layer

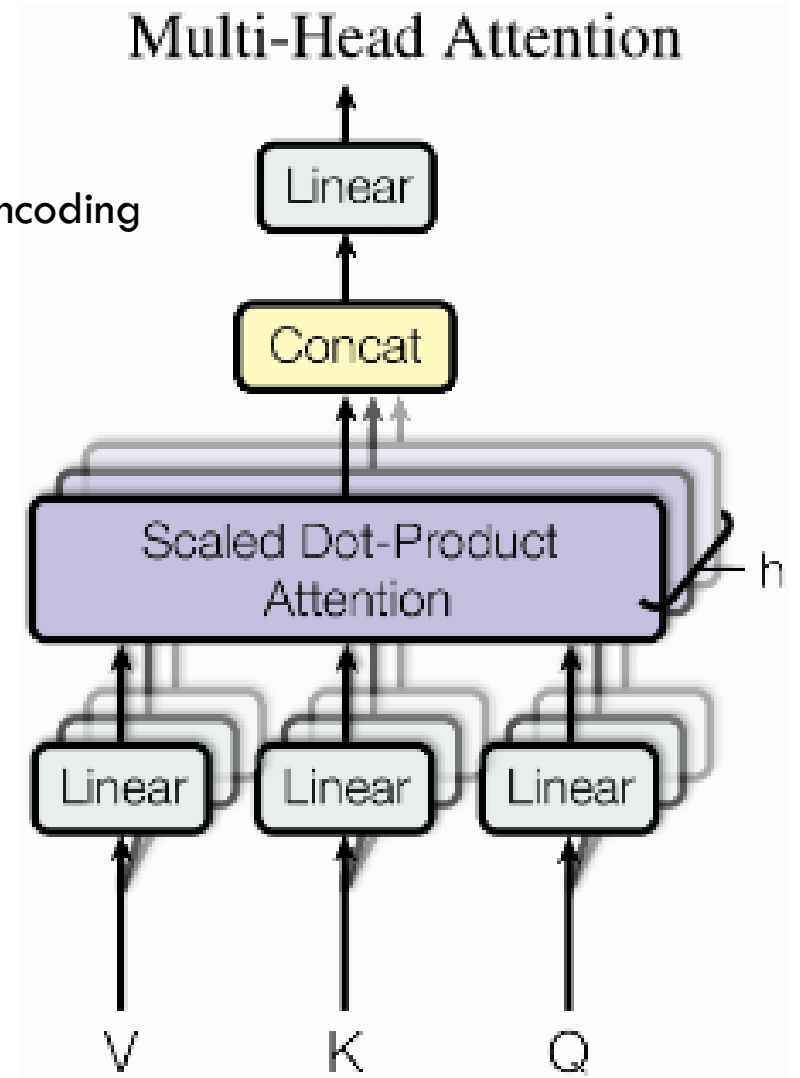


KEY QUERY VALUE INPUTS

- The Input to the MHA is patches projected through CNN + positional encoding



\equiv $X_inp(Seq, d_module)$ is duplicate into :



QUESTION #1

Images as input sequences of patches



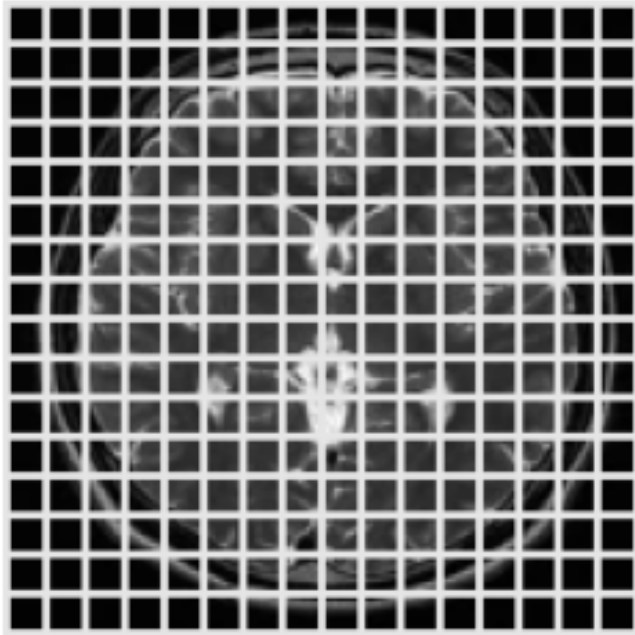
Bottom corners of image,
Denote black background ,
irrelevant for classification



**Are these 2 patches ideally
have the same Embedded
values ?**

QUESTION #1

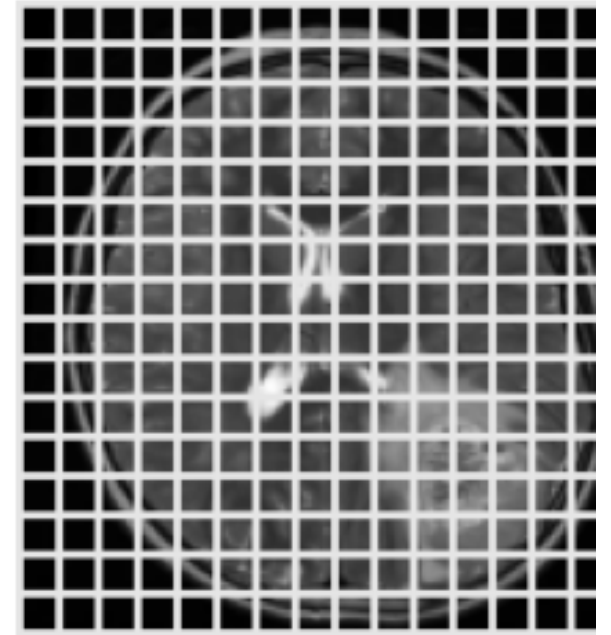
Images as input sequences of patches



Bottom corners of image,
Denote black background ,
irrelevant for classification

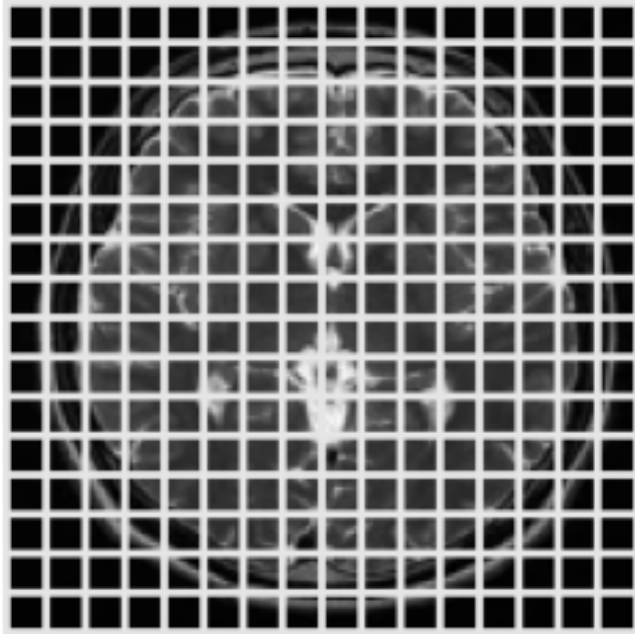
Ideally
Yes !

Are these 2 patches ideally
have the same Embedded
values ?



QUESTION #2

Images as input sequences of patches



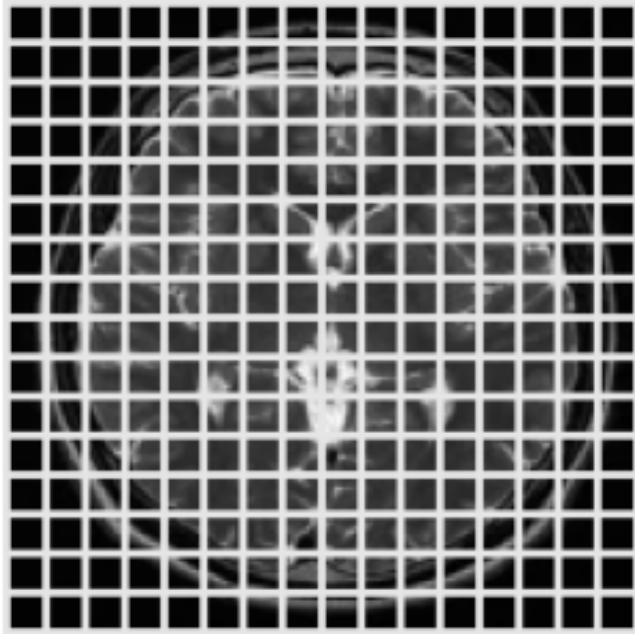
Bottom corners of image,
Denote black background ,
irrelevant for classification



**Are these 2 patches ideally
have the same K Q V values ?**

QUESTION #2

Images as input sequences of patches

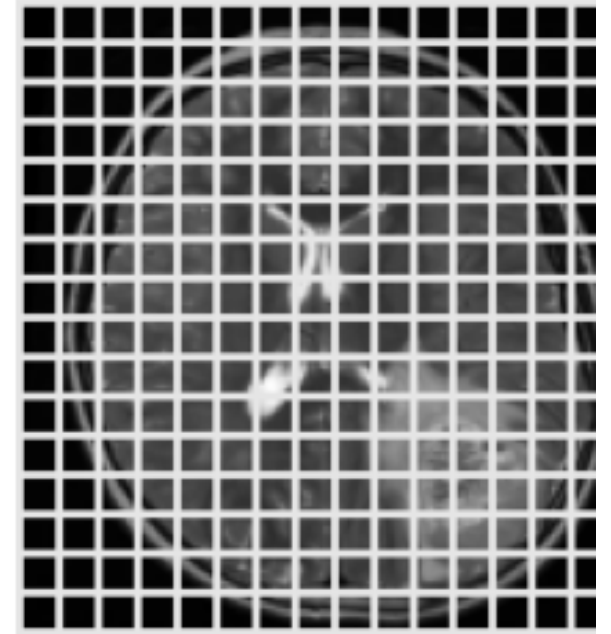


Bottom corners of image,
Denote black background ,
irrelevant for classification

NO !

**We added Positional
Encoding for sequence
distinction**

Are these 2 patches ideally have
the same K Q V values ?



SELF HEAD ATTENTION

Patch_inp \rightarrow input_Embedding + positional encoding \Rightarrow

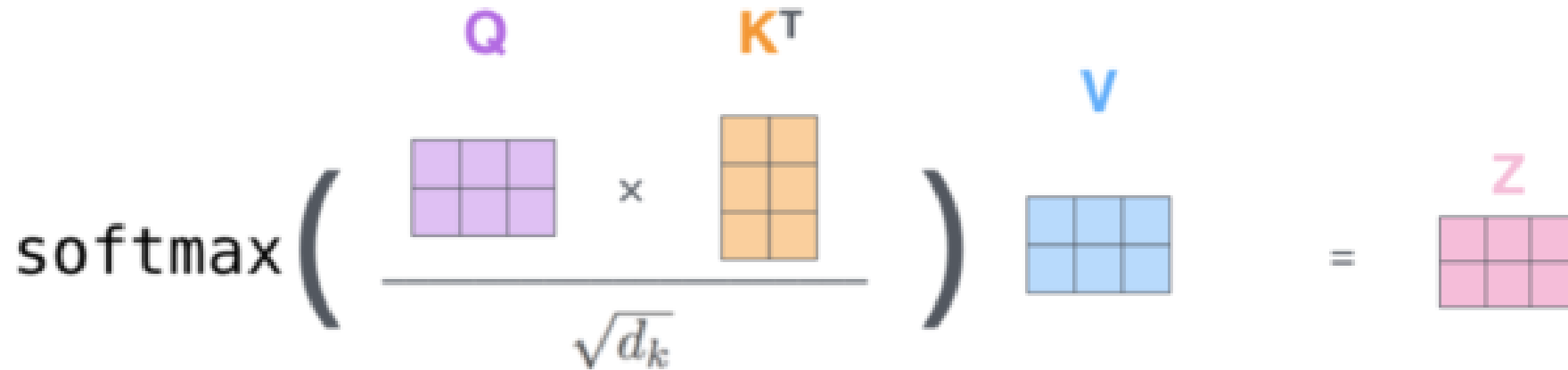
X_{inp} (Seq,d_module) is duplicate into :

Q (Seq,d_module)

K (Seq,d_module)

V (Seq,d_module)

Seq x Seq Attention map



The diagram illustrates the Self-Attention mechanism. It shows the input sequence X_{inp} being duplicated into three matrices: Q (purple, 2x3), K (orange, 3x2), and V (blue, 2x3). The Q and K matrices are combined to form a 'Seq x Seq Attention map' via a softmax operation. The result of this operation is then multiplied by the V matrix to produce the final Z matrix (pink, 2x3).

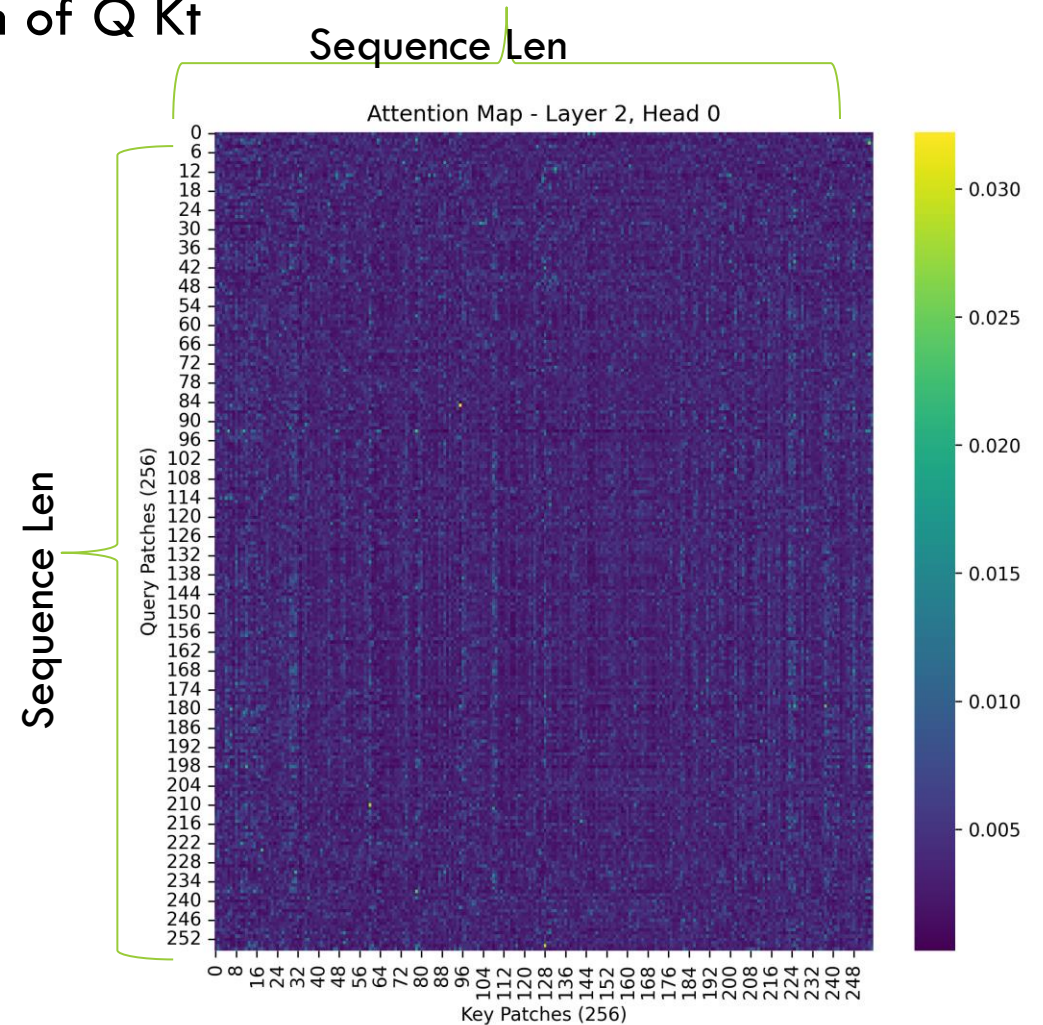
$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V = Z$$

Z matrix has:

1. Patches embedding info
2. PE info
3. Attention Matrix info

ATTENTION MAP

- Attention map is an output of the Softmax activation of $Q K^T$
- It has $[\text{Sequence_Len} \times \text{Sequence_Len}]$ dimensions
- Each cell denote the attention A_{ij} between 2 patches
- Sum of columns provide patch importance



How to compute Self-Attention?

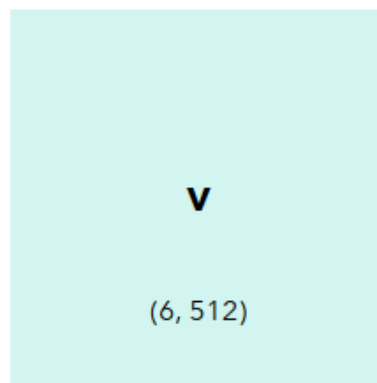
Example from NLP area:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

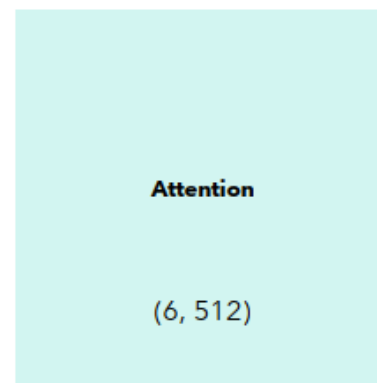
	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

(6, 6)

X



=

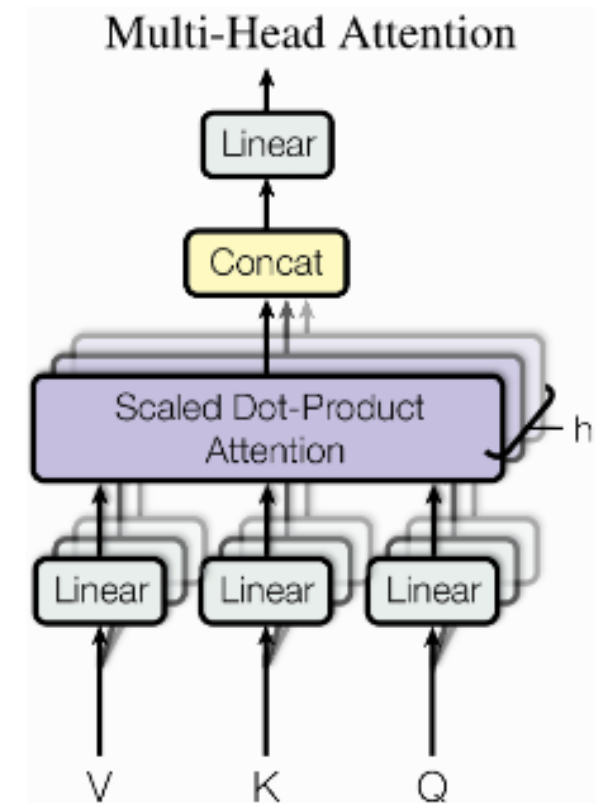


Each row in this matrix captures not only the meaning (given by the embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other words.

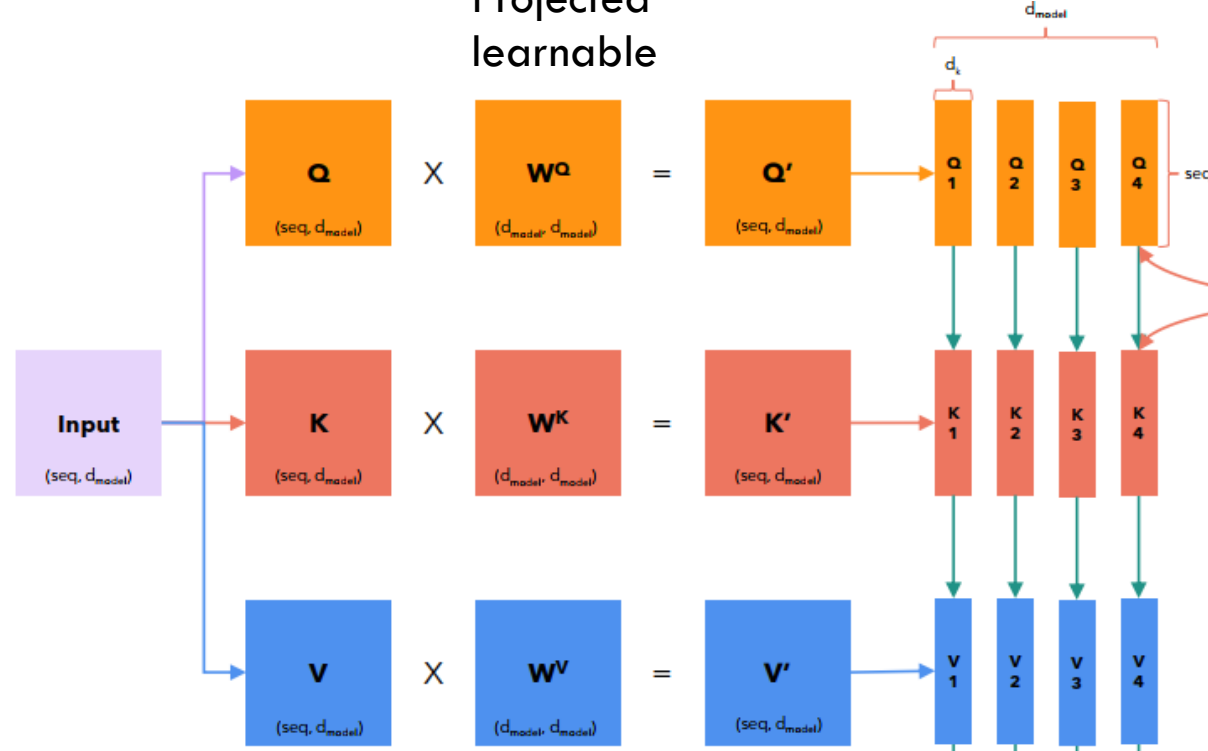
MULTI HEAD ATTENTION

By means of Multi Head Attention the encoder issue for each image patch the following characteristics:

- embedded value as issued from CNN
- Relative Patch position in sequence by PE
- Interactions with other patches given by attention matrix

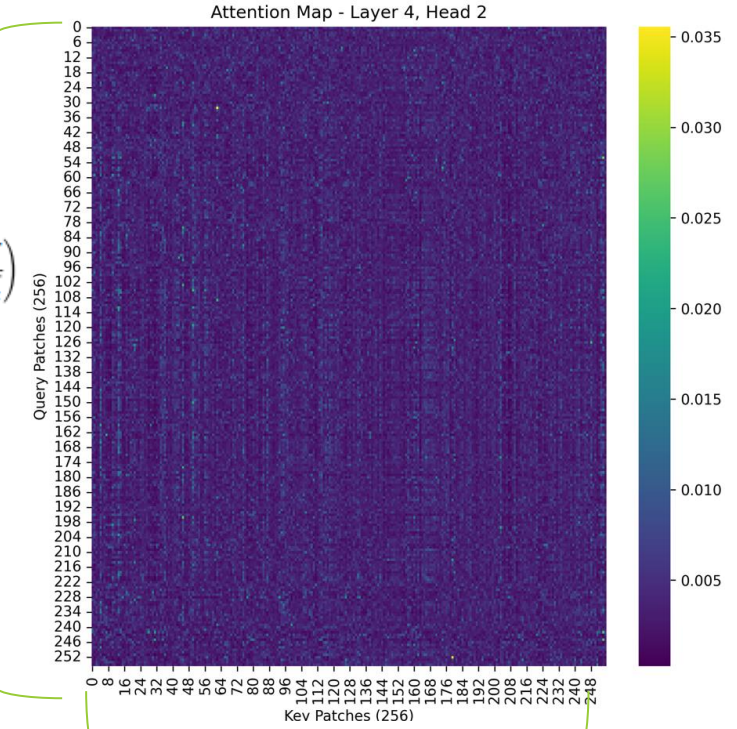
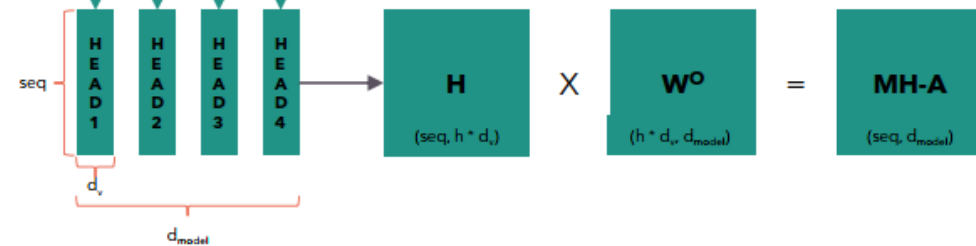


Projected learnable



$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$



- seq = sequence length
- d_{model} = size of the embedding vector
- h = number of heads

TRAINING NETWORK...

Based on 253 images :

train_loss= 0.00429

train_acc=1.000

val_loss= 1.910

val_acc=0.692

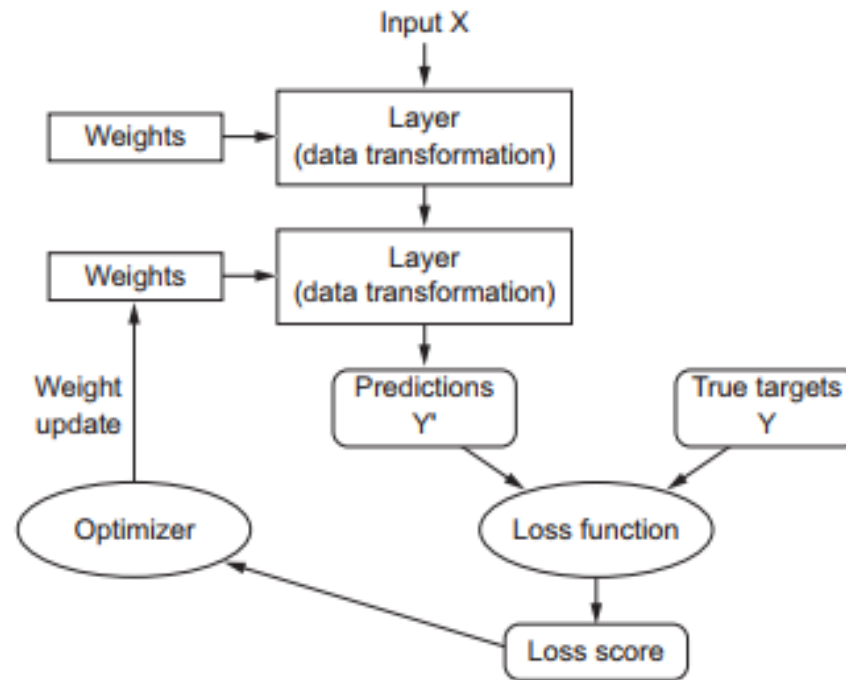


Figure 1.9 The loss score is used as a feedback signal to adjust the weights.

TRAINING NETWORK

Based on 253 images :

train_loss= 0.00429

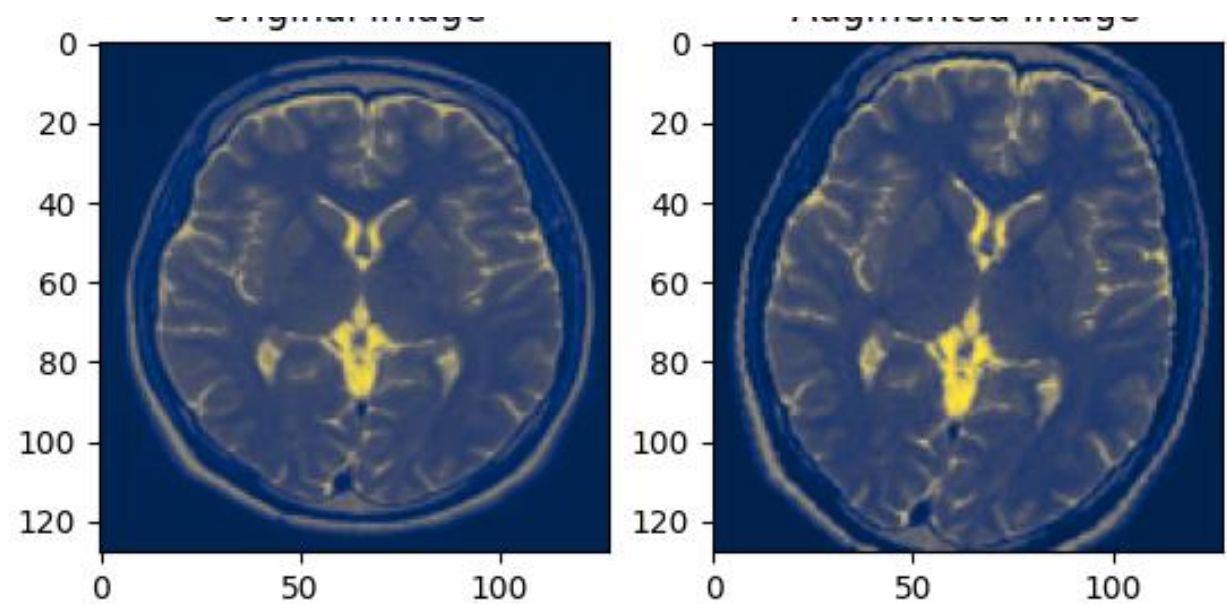
train_acc=1.000

val_loss= 1.910

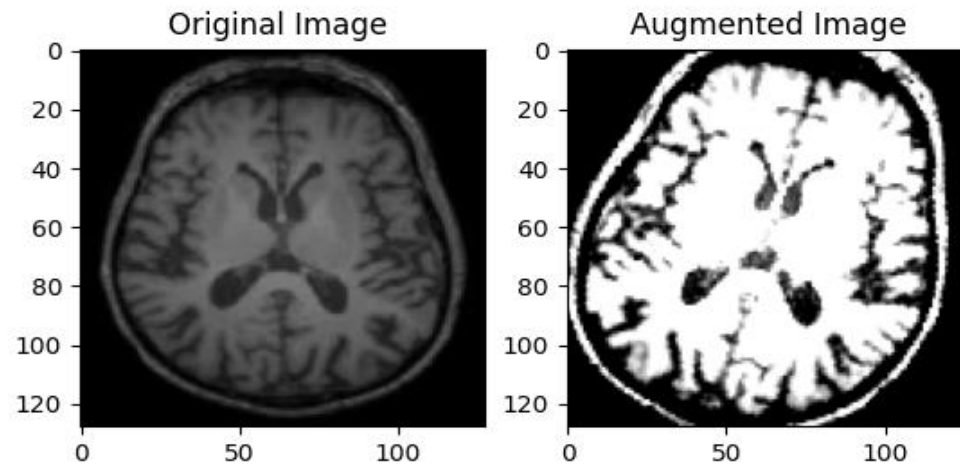
val_acc=0.692

Overfit

ADD DATA AUGMENTATION



- Enlarge our dataset to 512 images
 - Shrinking
 - Stretching
 - Rotating
 - Flipping
 - Cropping
 - Normalizing
 - Adding noises
 - Brightness / contrasts / Gamma corrections



AFTER DATA AUGMENTATION

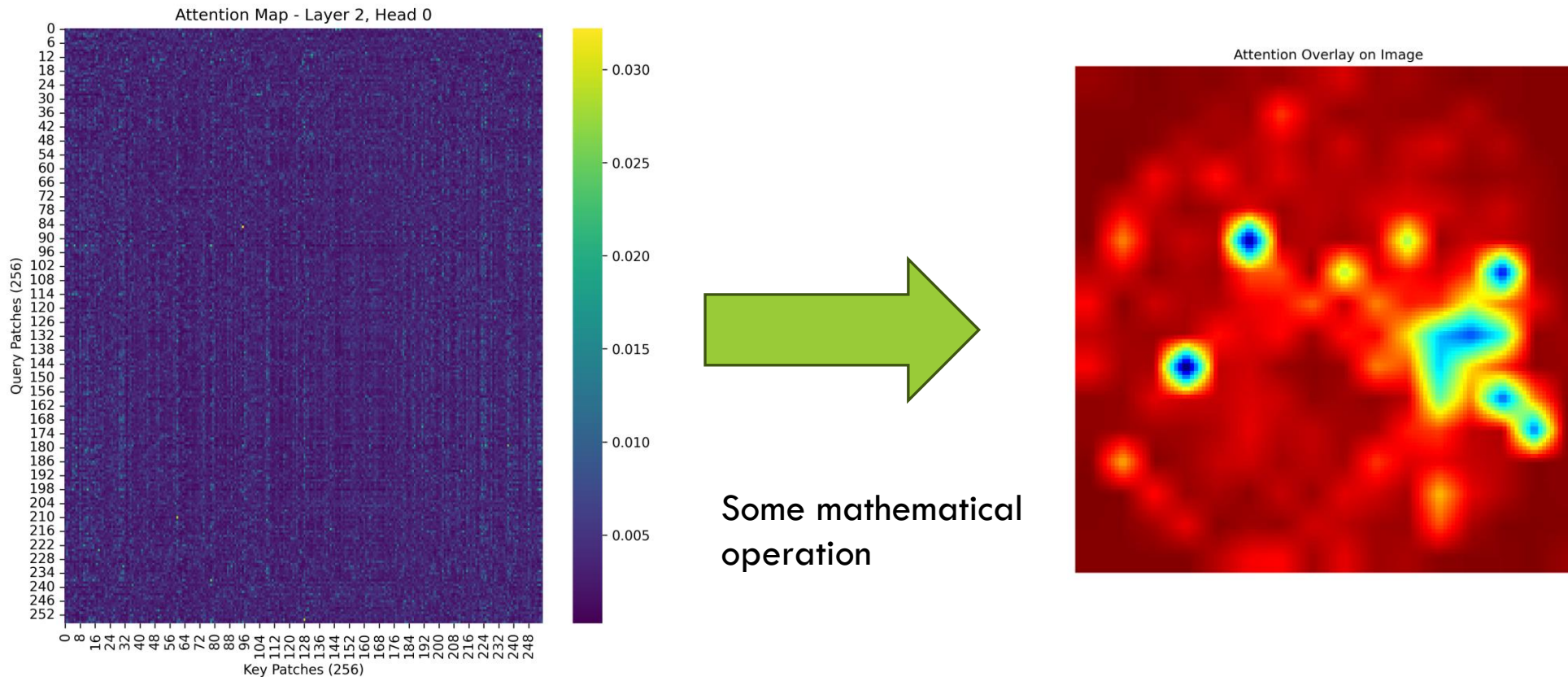
ViT results {'test': 0.9230769276618958, 'val': 1.0}

The good news :

- Our model pinpoint the tumor in more than 90 % accuracy
- No overfit
- No false positives of brain lobes as tumors

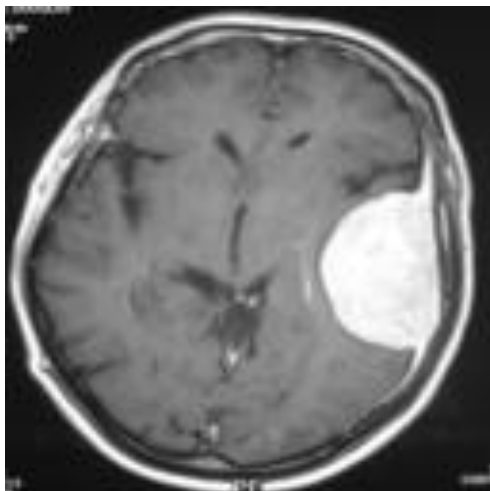
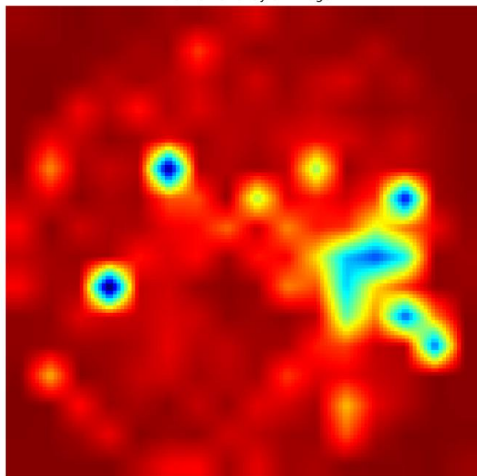
ATTENTION MAP TO HEATMAP AS DEBUG HOOK

- Heatmap pinpoint the more important areas in the tested image
- Helps the developer to figure out if hir model focus the right areas in image

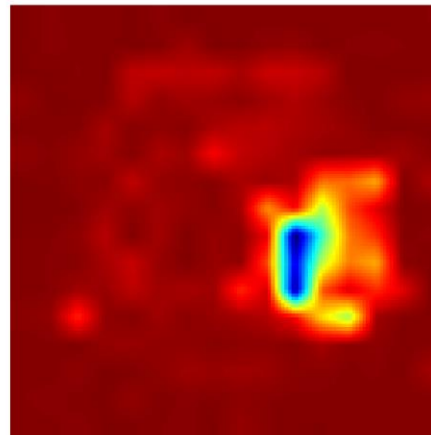


VISUALIZE HEATMAP

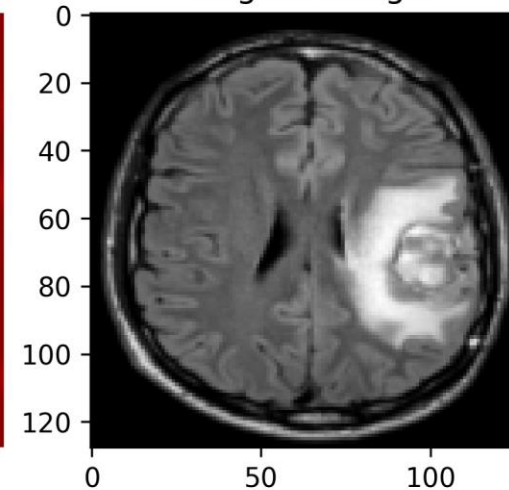
Attention Overlay on Image



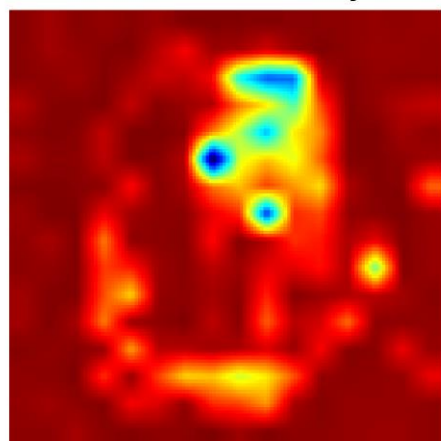
Attention Overlay



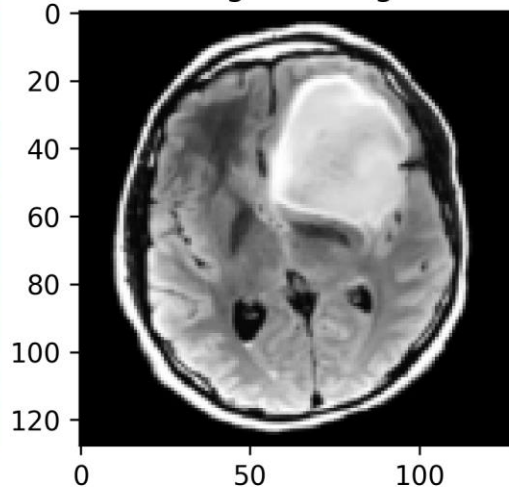
Original Image



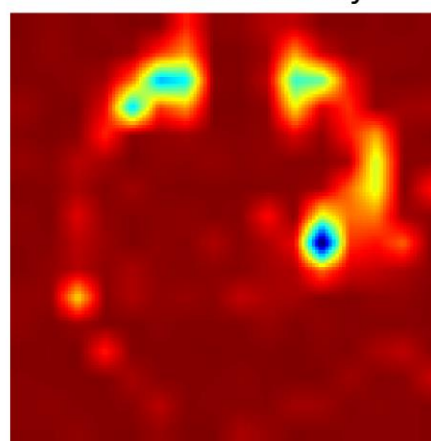
Attention Overlay



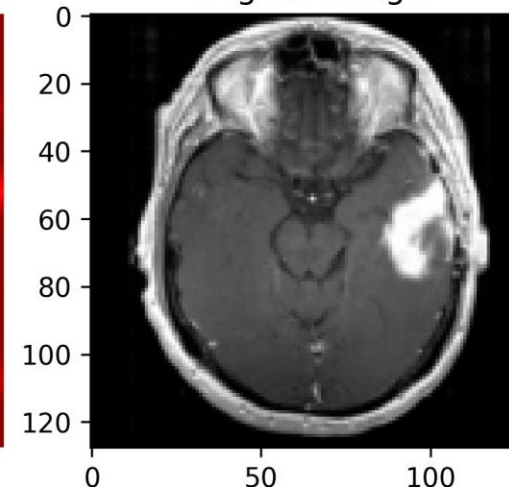
Original Image



Attention Overlay

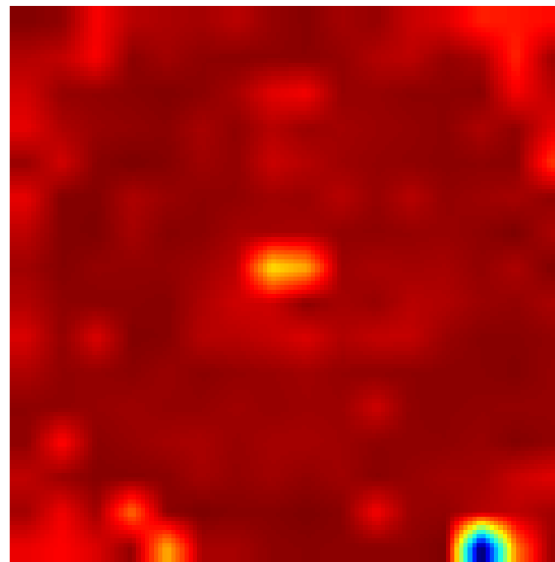


Original Image



ATTENTION MAP

Attention Overlay



Original Image

