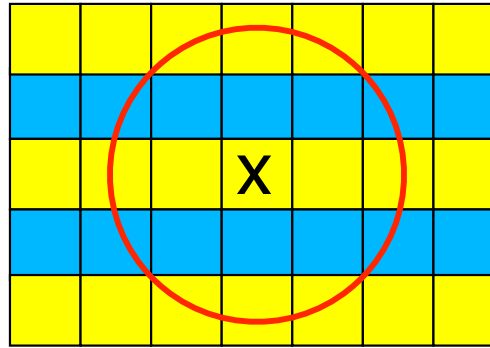


# Transformers

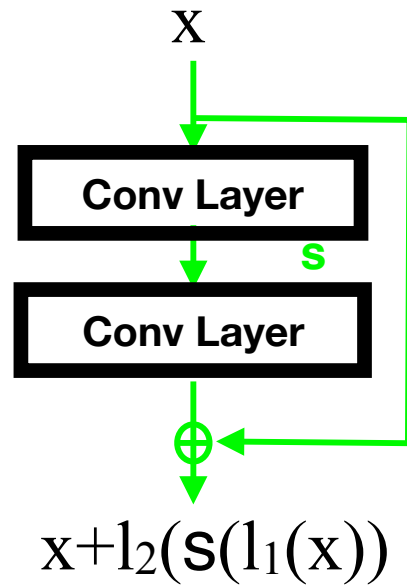
Pascal Fua  
IC-CVLab

# Reminder: Image Specificities

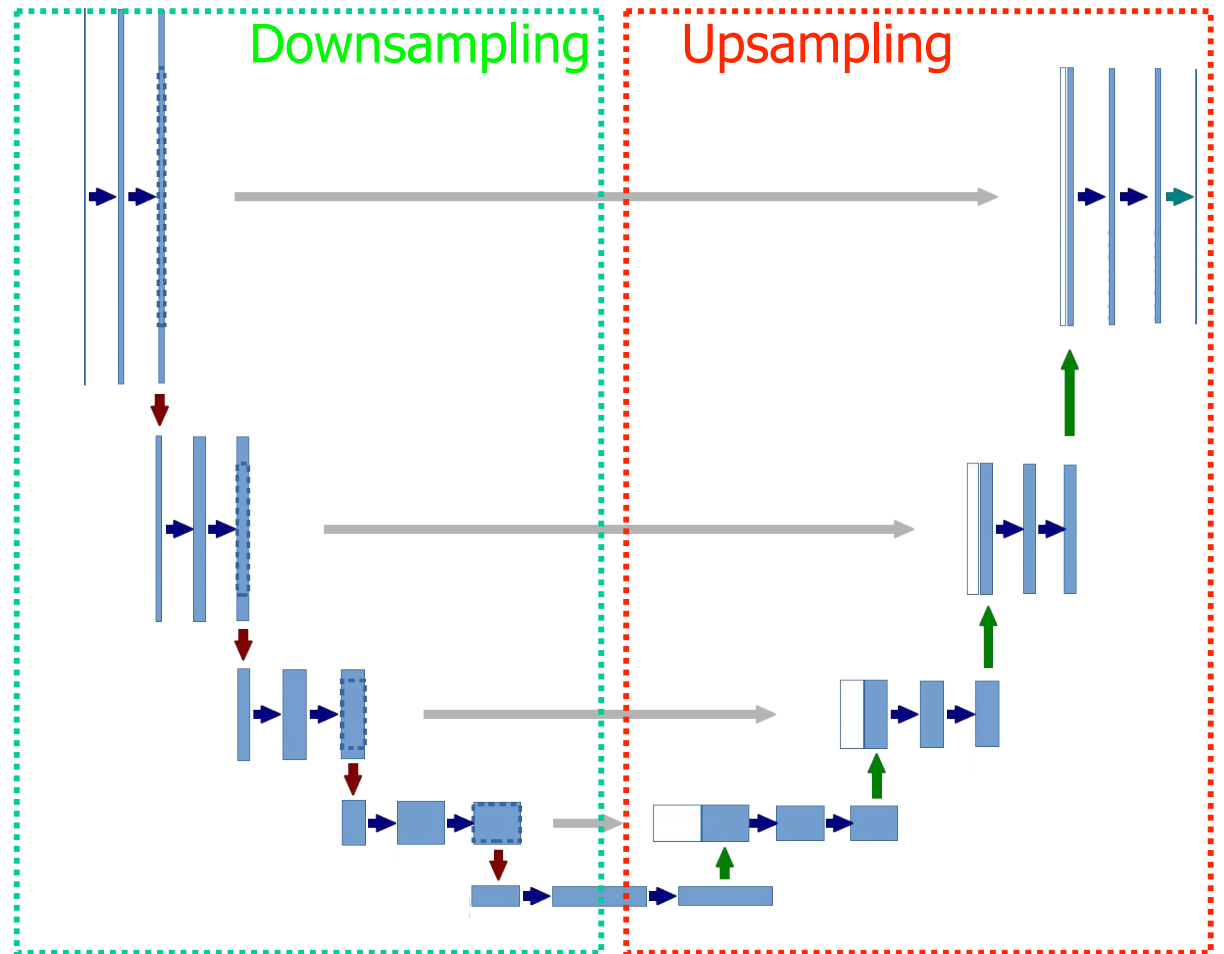
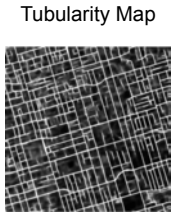
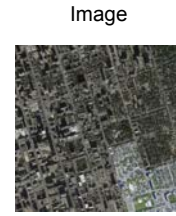


- In a typical image, the values of **neighboring pixels** tend to be more highly correlated than those of distant ones.
  - An image filter should be translation equivariant.
- > These two properties can be exploited to drastically reduce the number of weights required by CNNs using so-called **convolutional** layers.

# Reminder: ResNet to U-Net



ResNet block



U-Net

EPFL  $\rightarrow$  Long range connections are handled via downsampling.

# Natural Language Example

The restaurant refused to serve me a ham sandwich, because **it** only cooks vegetarian food. In the end, they just gave me **two** slices of bread. Their ambience was just as **good** as the food and service.

Given your preferences should you go this restaurant?

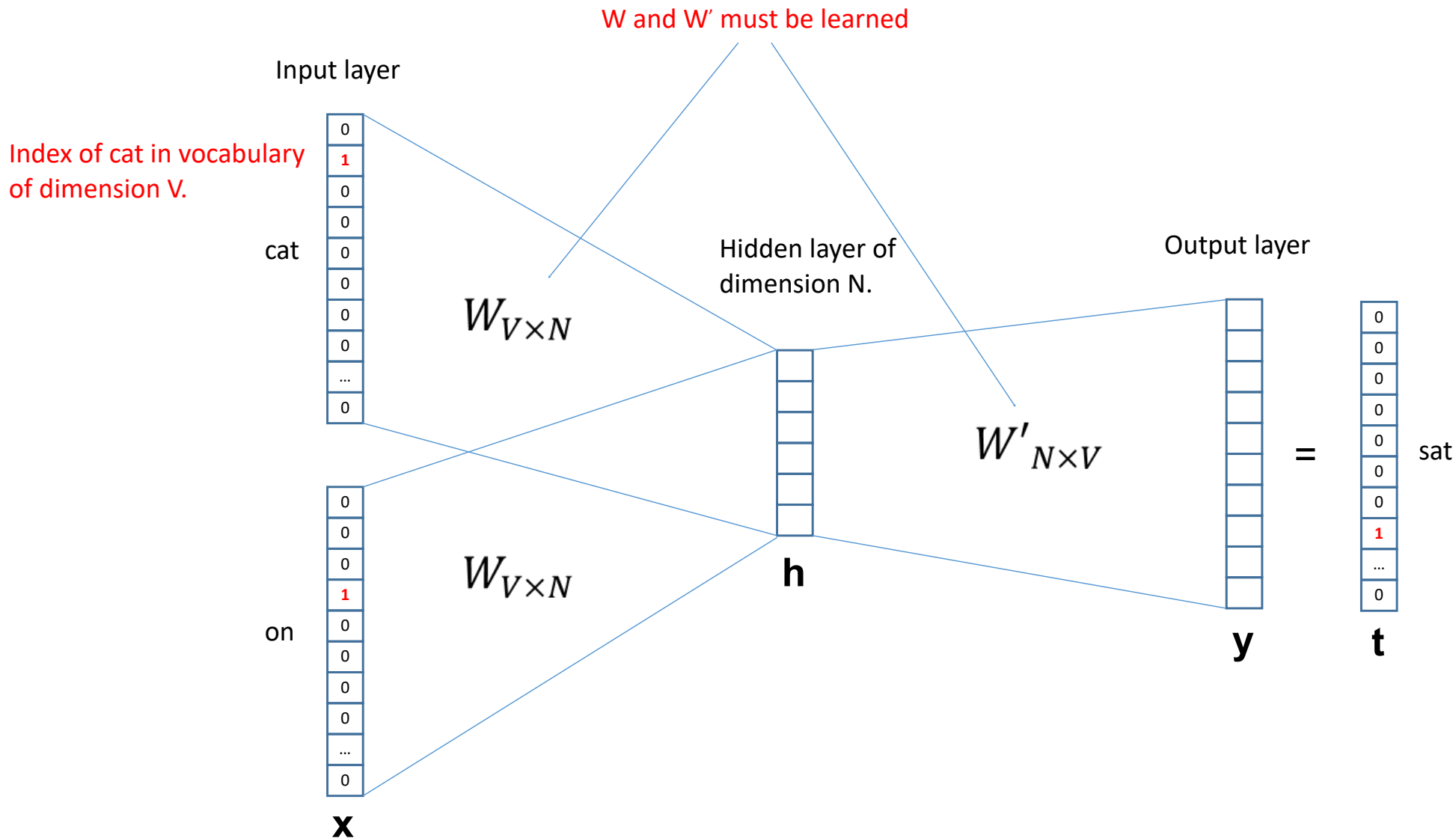
Seriously?

What does "it" refer to?

# Context Matters

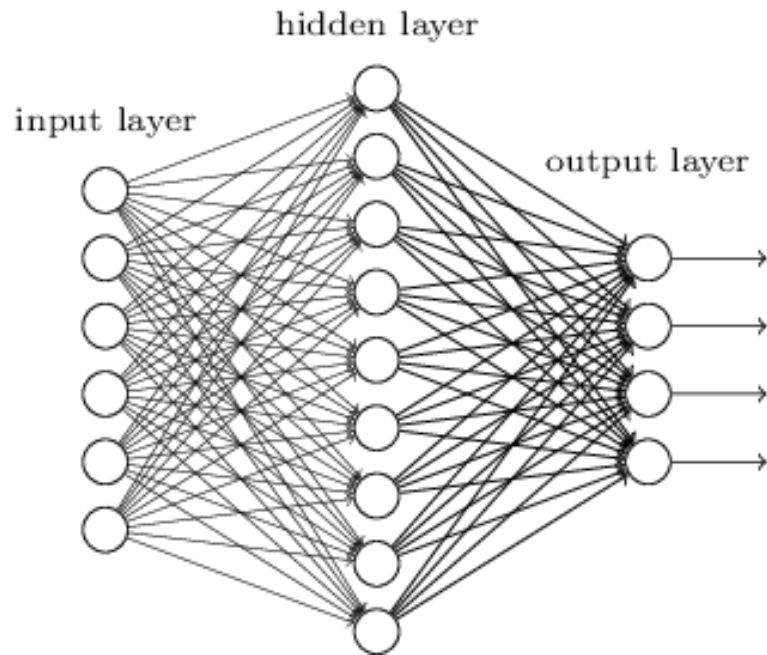
- Meaning cannot be had from the syntax alone!
  - Some words are given “attention” by others.
  - Such words are not necessarily close to each other.
- > Assuming that words have been converted to vectors, transformers have been invented to deal with this issue.

# Reminder: Word2Vec



Once the training is complete, the  $W$  matrices associate to each word a vector of dimension  $N$ .

# Fully Connected Layers Revisited



Standard fully connected layer

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$= \sigma(\tilde{\mathbf{W}}\tilde{\mathbf{x}})$$

➔ We can drop the bias terms.

# Self Attention

Given  $I$  words  $\mathbf{x}_i$ , compute the self attention

$$\forall i, sa[\mathbf{x}_i] = \sum_{j=1}^I a[\mathbf{x}_i, \mathbf{x}_j] \mathbf{W}_v \mathbf{x}_j$$

Attention given to  $\mathbf{x}_i$  by all  $\mathbf{x}_j$

Attention given by  $\mathbf{x}_j$  to  $\mathbf{x}_i$

Value of  $\mathbf{x}_j$



# Self Attention in Matrix Form (1)

Given  $I$  words  $\mathbf{x}_i$ , let  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_I]$



Query:  $\mathbf{X}_q = \mathbf{X}\mathbf{W}_q$

Key:  $\mathbf{X}_k = \mathbf{X}\mathbf{W}_k$

Value:  $\mathbf{X}_v = \mathbf{X}\mathbf{W}_v$

# Self Attention in Matrix Form (2)

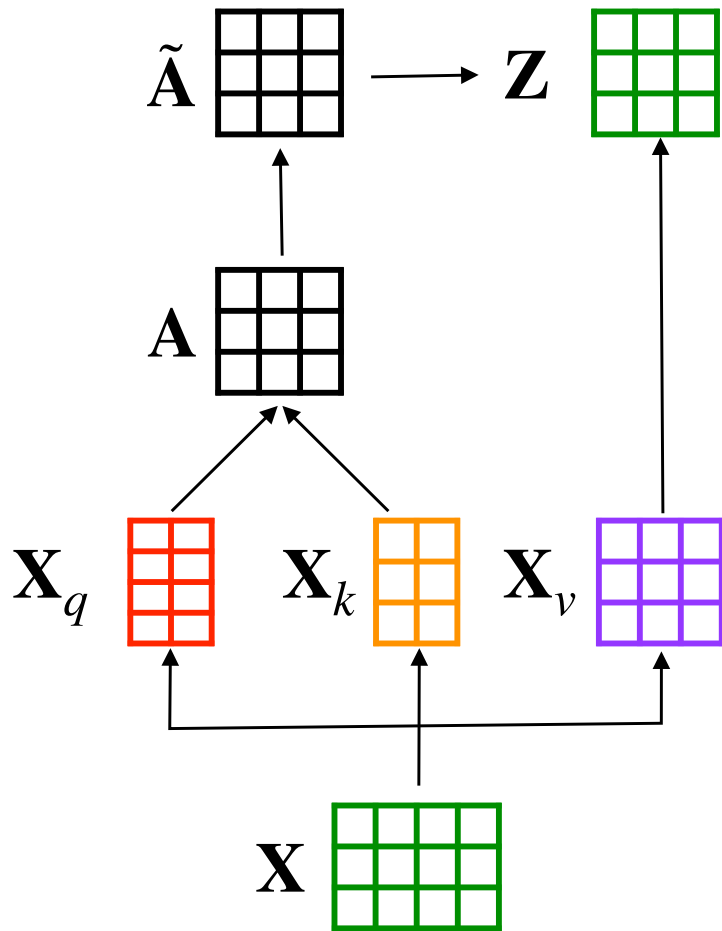
$$\text{softmax} \left( \frac{\begin{matrix} \mathbf{Q} \\ \begin{matrix} \square & \square & \square \\ \square & \square & \square \end{matrix} \end{matrix} \times \begin{matrix} \mathbf{K}^T \\ \begin{matrix} \square & \square \\ \square & \square \\ \square & \square \end{matrix} \end{matrix}}{\sqrt{d_k}} \right) \begin{matrix} \mathbf{V} \\ \begin{matrix} \square & \square & \square \\ \square & \square & \square \end{matrix} \end{matrix}$$
$$= \begin{matrix} \mathbf{Z} \\ \begin{matrix} \square & \square & \square \\ \square & \square & \square \end{matrix} \end{matrix}$$

$$\mathbf{A} = \frac{1}{\sqrt{d_k}} \mathbf{X}_q \mathbf{X}_k^T$$
$$= \frac{1}{\sqrt{d_k}} \mathbf{X} \mathbf{W}_q \mathbf{W}_k^T \mathbf{X}$$

$$\tilde{\mathbf{A}}_t = \text{softmax}(\mathbf{A}_{t,:})$$

$$\mathbf{Z} = \tilde{\mathbf{A}} \mathbf{X}_v$$

# Self Attention in Matrix Form (3)



$$\mathbf{X}_k = \mathbf{X}\mathbf{W}_k$$

$$\mathbf{X}_v = \mathbf{X}\mathbf{W}_v$$

$$\mathbf{X}_q = \mathbf{X}\mathbf{W}_q$$

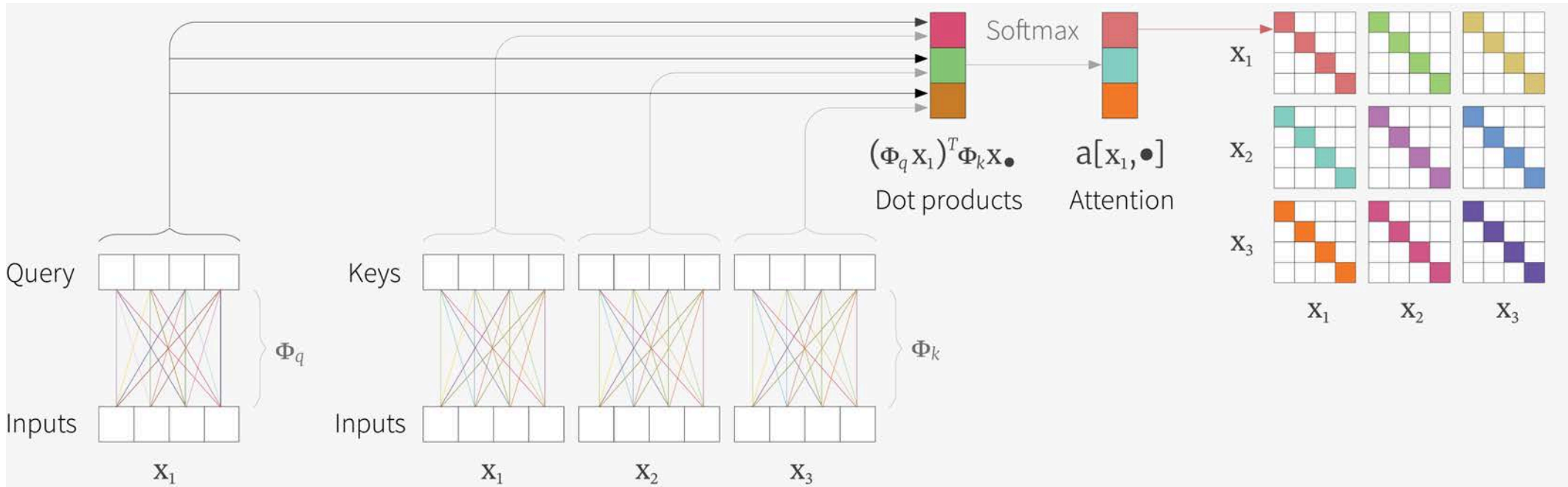
$$\begin{aligned} \mathbf{A} &= \frac{1}{\sqrt{d_k}} \mathbf{X}_q \mathbf{X}_k^T \\ &= \frac{1}{\sqrt{d_k}} \mathbf{X} \mathbf{W}_q \mathbf{W}_k^T \mathbf{X} \end{aligned}$$

$$\tilde{\mathbf{A}}_t = \text{softmax}(\mathbf{A}_{t,:})$$

$$\mathbf{Z} = \tilde{\mathbf{A}} \mathbf{X}_v$$

$$Sa(\mathbf{X}) = \mathbf{Z} = \text{softmax}[\mathbf{X}\mathbf{W}_q \mathbf{W}_k^T \mathbf{X}^T] \mathbf{X}\mathbf{W}_v$$

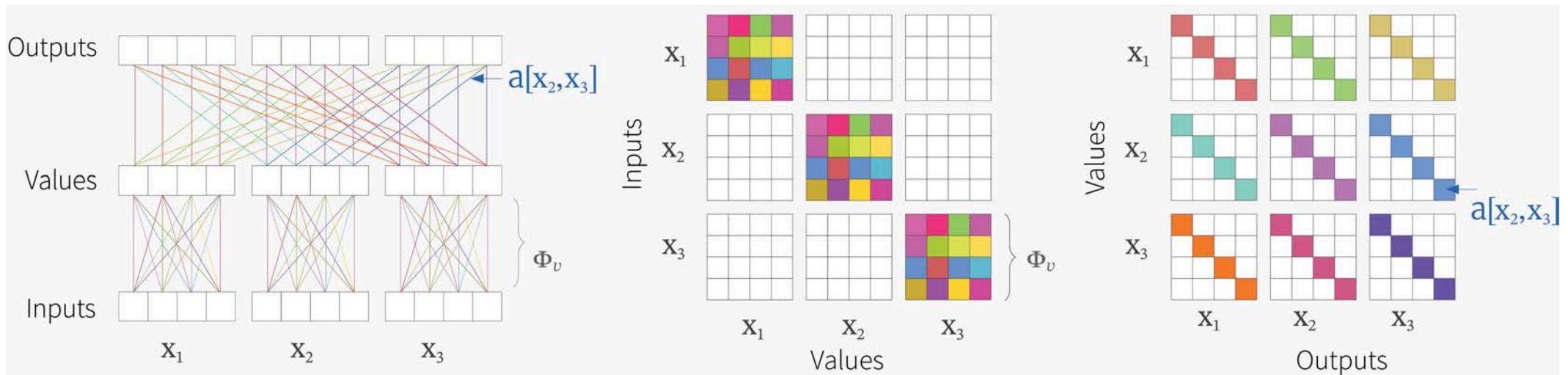
# Dot-Product Self Attention



$$\forall i, sa[\mathbf{x}_i] = \sum_{j=1}^I a[\mathbf{x}_i, \mathbf{x}_j] \mathbf{W}_v \mathbf{x}_j$$

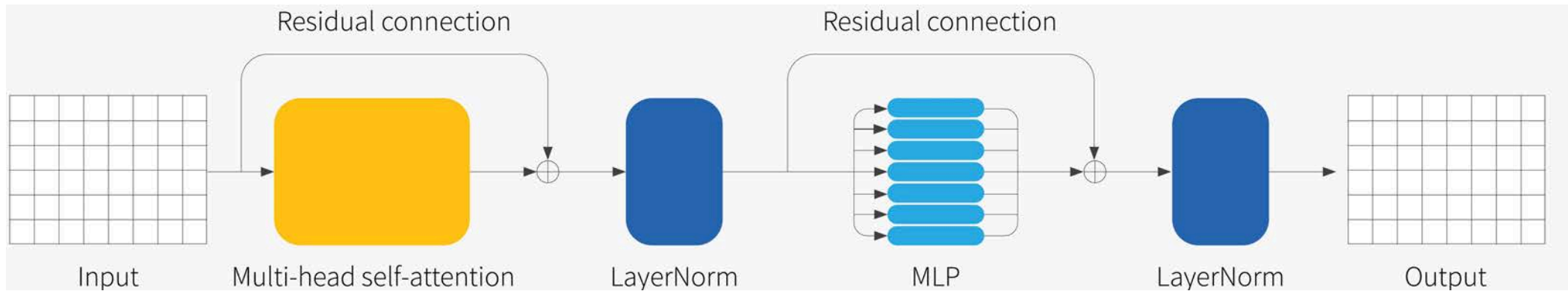
$$a[\mathbf{x}_i, \mathbf{x}_j] = \frac{\exp[(\mathbf{W}_q \mathbf{x}_i)^T \mathbf{W}_k \mathbf{x}_j]}{\sum_{j=1}^I \exp[(\mathbf{W}_q \mathbf{x}_i)^T \mathbf{W}_k \mathbf{x}_j]}$$

# Sparse Attention Weights



Given  $\mathbf{X} = [x_1, \dots, x_I]$ , we compute  $Sa(\mathbf{X})$  using far fewer weights than if we used a fully connected network.

# Transformer Layer



$$\mathbf{X} \leftarrow \mathbf{X} + Sa(\mathbf{X})$$

$$\mathbf{X} \leftarrow LayerNorm(\mathbf{X})$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + mlp[\mathbf{x}_i] \quad \forall i$$

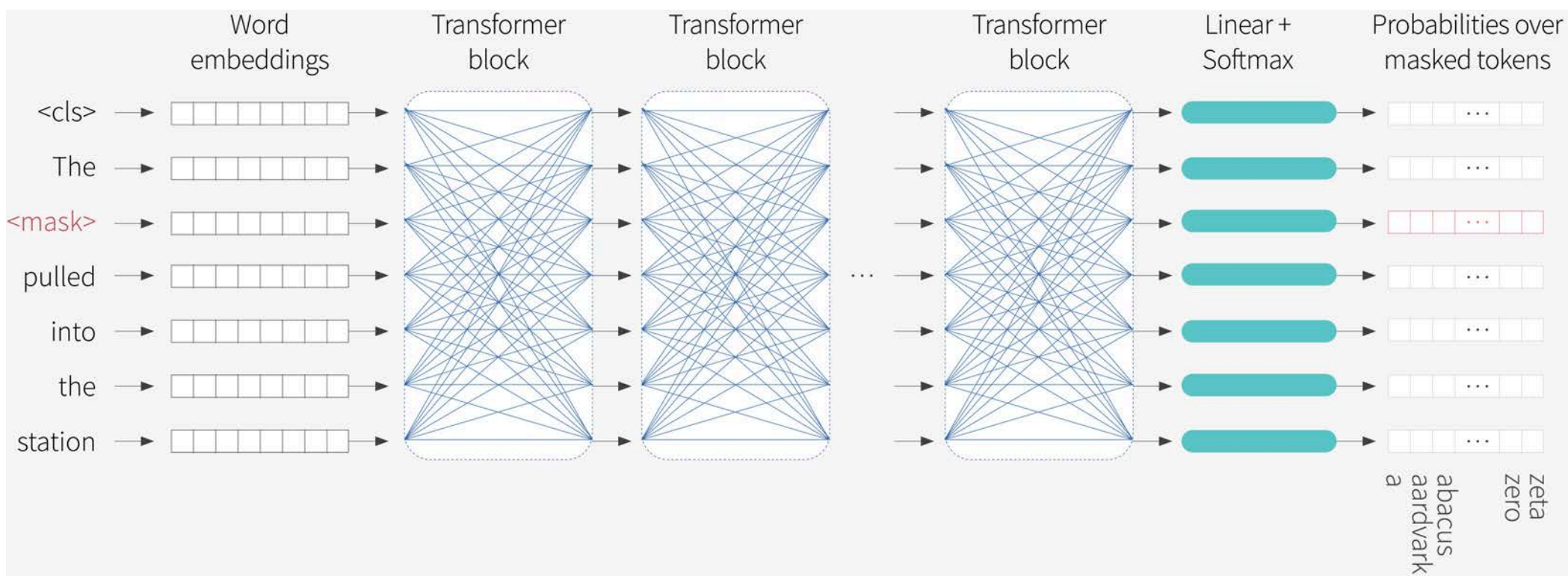
$$\mathbf{X} \leftarrow LayerNorm(\mathbf{X})$$

# Basic PyTorch Code

```
class SANet(nn.Module):  
    def __init__(self, kDim, dTok, sigm=1e-6):  
        self.Wv = torch.nn.Parameter(sigm*torch.randn(dTok, dTok))  
        self.Wk = torch.nn.Parameter(sigm*torch.randn(dTok, kDim))  
        self.Wq = torch.nn.Parameter(sigm*torch.randn(dTok, kDim))  
        # Allocate the 3 matrices  
  
    def forward(self, xt):  
        xk = torch.matmul(xp, self.Wk)  
        xq = torch.matmul(xp, self.Wq)  
        xv = torch.matmul(xt, self.Wv)  
        # Compute key, query, and value  
  
        A = torch.matmul(xq, xk.transpose(1, 2))  
        A = F.softmax(A, dim=2)  
        # Compute the self attention  
        A = torch.matmul(A, xv)  
  
        return (A)
```

-> The heart of chatGPT!

# Optional: Bidirectional Encoder Representations from Transformers (BERT)

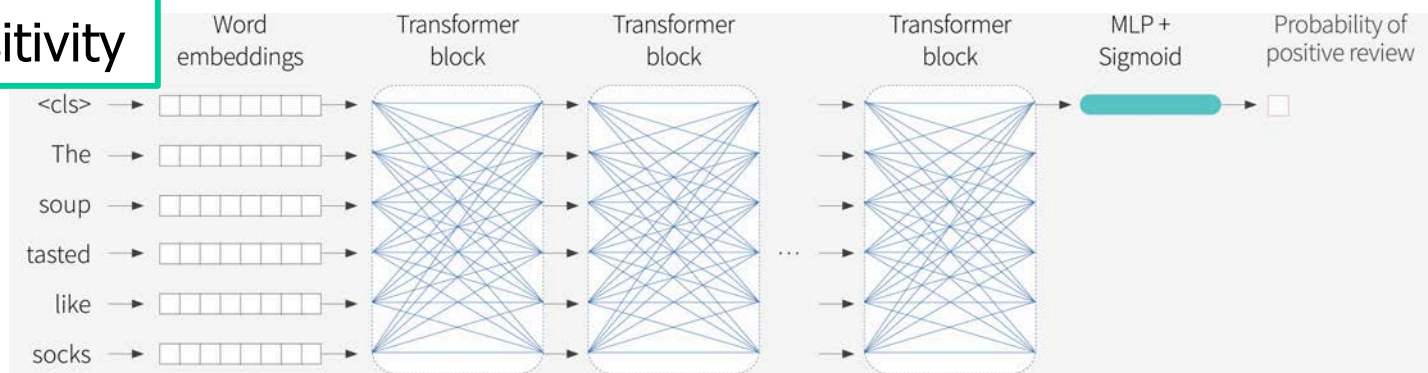


- A sequence of transformer blocks is trained to predict a missing word in a sentence.
- This forces the transformer to learn something about the syntax of the language.

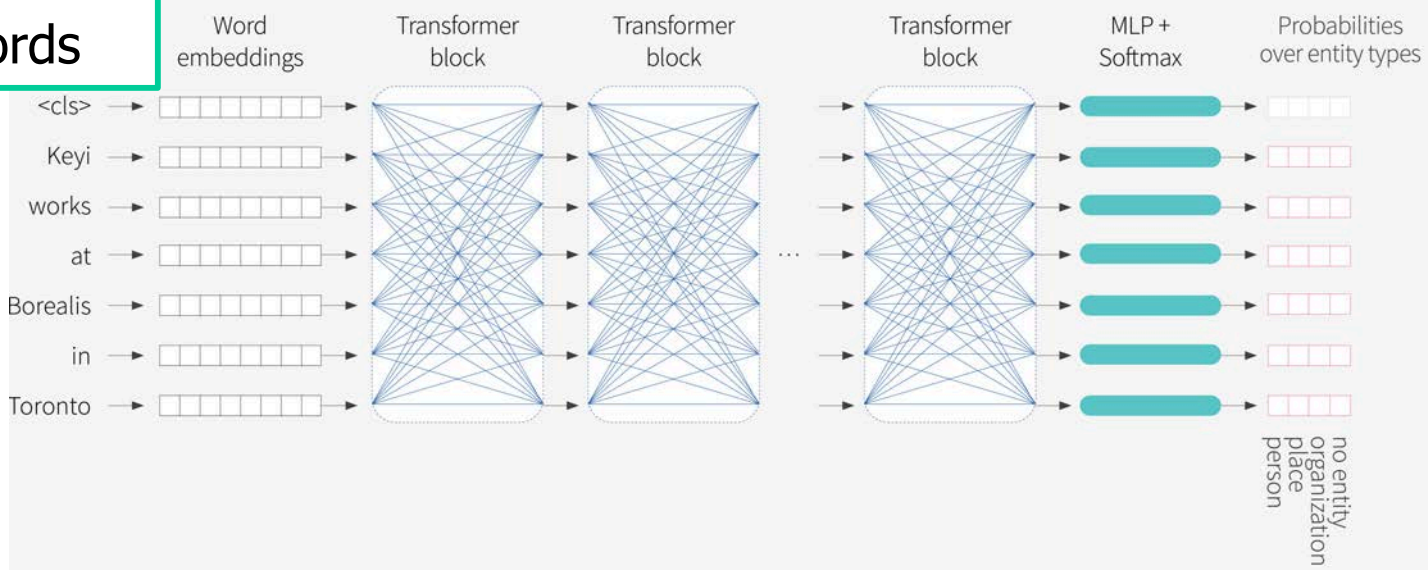


# Optional: Bidirectional Encoder Representations from Transformers (BERT)

Rate the positivity

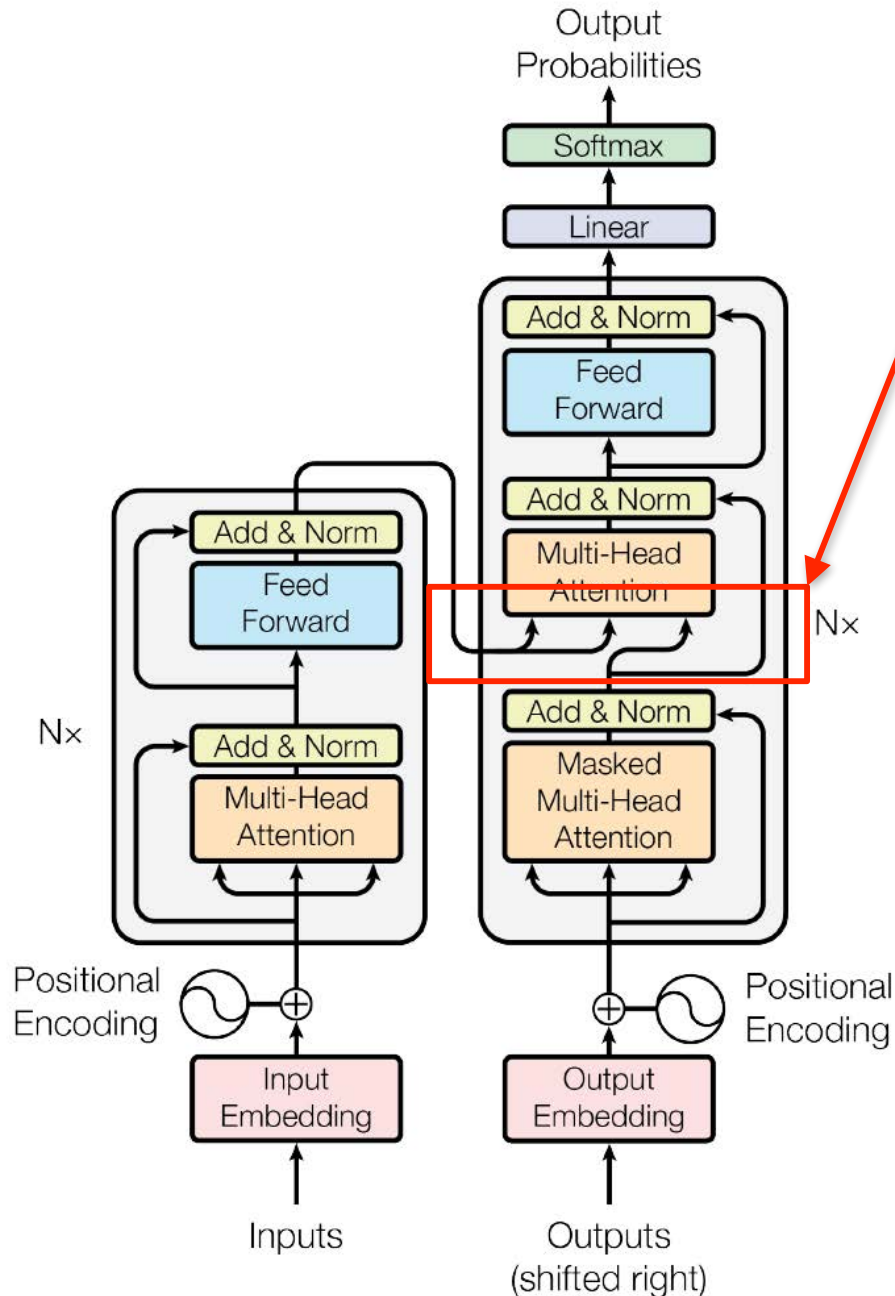


Classify words



Given additional training data, the representation learned in this way can be used to perform various tasks.

# Optional: ChatGPT



- At this point, the transformer layer is fed both the prompt and the already generated text.
- It uses this information to guess the next word.
- The process is then iterated.

Keys to ChatGPT success:

- The network looks back as far as needed.
- It uses a huge corpus.
- Human guided training.

# Optional: ChatGPT

PV

How do I recover from a stall



If you are asking about a stall in an aircraft, the first thing to do is to identify the cause of the stall. This may involve checking the flight instruments, looking outside the aircraft for visual clues, and consulting the aircraft's flight manual. Once the cause of the stall has been determined, the appropriate recovery procedure can be initiated. This may involve reducing the angle of attack, increasing power, or taking other corrective action to restore the flow of air over the wings and regain control of the aircraft. It is important to follow the correct procedures for recovering from a stall, as attempting to recover improperly can make the situation worse. If you are not trained in how to recover from a stall, it is best to consult with a qualified instructor or pilot.

A little late for that.

PV

How do I recover from a spin in a glider



Recovering from a spin in a glider involves a few key steps. First, you should try to stay calm and avoid panicking. This will help you think clearly and make the right decisions to recover from the spin.

Next, you should locate the spin recovery instructions in your glider's flight manual and follow them carefully. These instructions will be specific to your glider and will outline the correct procedure for recovering from a spin. In general, the steps for recovering from a spin in a glider will involve using the controls to stop the rotation of the aircraft and regain control.

That's going to get you killed!!

Here are some general guidelines for recovering from a spin in a glider:

1. Pull the control stick firmly and fully back to the neutral position. This will stop the rotation of the aircraft and help you regain control.
2. Apply full opposite rudder to the direction of the spin. This will help stop the rotation of the aircraft and straighten it out.
3. Once the rotation has stopped and the aircraft is flying straight and level again, gently ease the control stick forward to return to normal flight.

It's important to remember that recovering from a spin in a glider can be a challenging and potentially dangerous maneuver. If you're not sure how to recover from a spin, or



To recover from a stall in aviation, the pilot should follow these steps:

1. Reduce the angle of attack by pushing forward on the control column or yoke.
2. Apply full power to the engine(s).
3. Keep the wings level by using the ailerons or by turning the yoke in the direction opposite the wing that is dropping.
4. Once the airplane begins to pick up speed and the wing starts to produce lift again, gently pull back on the control column or yoke to level the wings and regain altitude.
5. Be aware of the altitude and terrain and take appropriate action to regain altitude if necessary.
6. Once the aircraft is out of stall, maintain a safe airspeed, and climb to a safe altitude.

It's important to note that these steps are general guidelines and may vary depending on the type of aircraft and the specific circumstances of the stall. It's also important to practice stall recovery in a simulated environment before attempting it in an actual flight. **It is.**



Recovering from a spin in a glider typically involves the following steps:

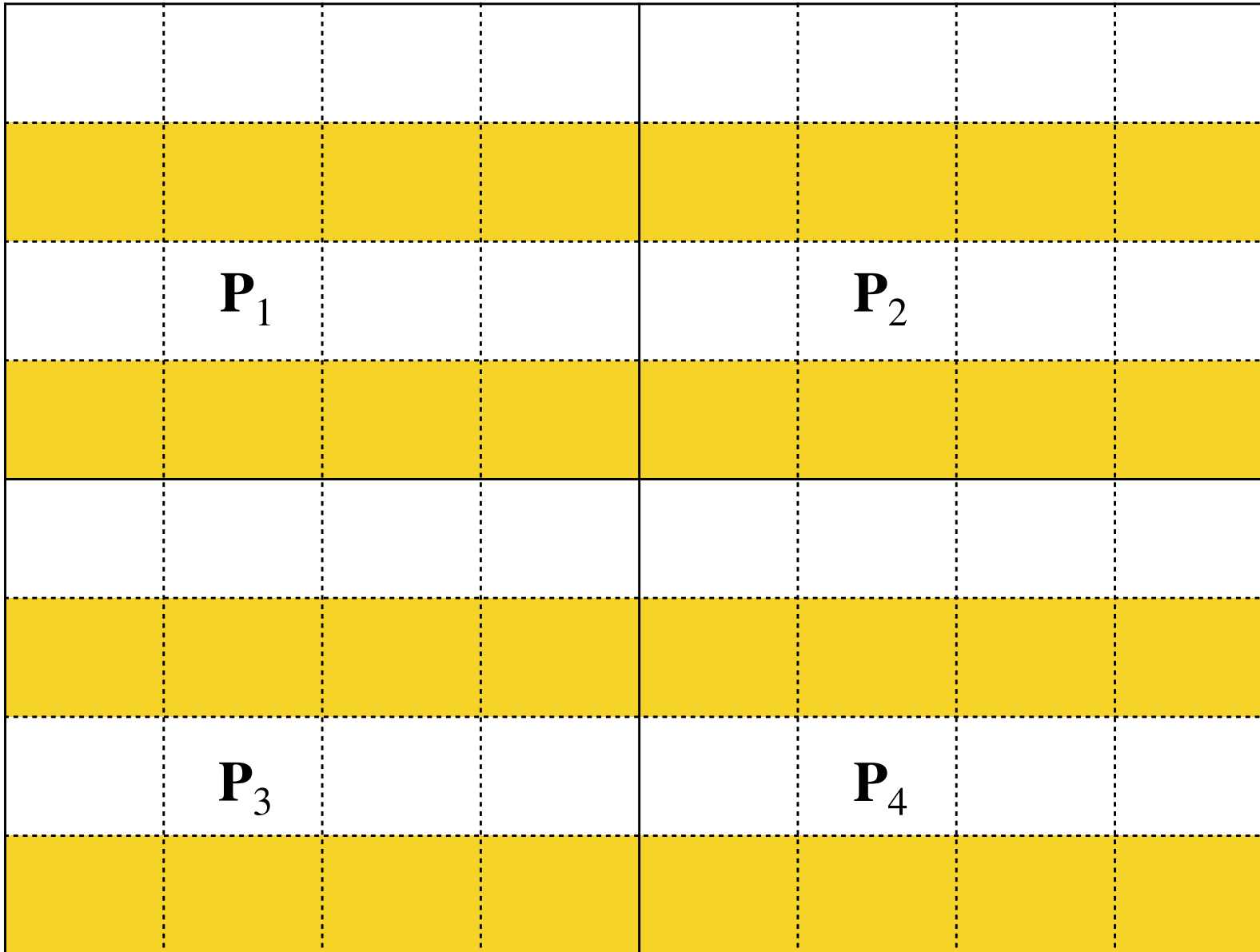
1. Identify the spin and its direction. **Correct. You will live.**
2. Neutralize the controls by releasing any pressure on the ailerons, rudder and elevator.
3. Apply full opposite rudder to the spinning direction and hold it until rotation stops.
4. Apply forward stick to reduce the angle of attack and increase airspeed.
5. Level the wings and check for proper control response.

It's important to note that a spin recovery procedure may vary depending on the type of glider and the specific training you have received. Always follow the instructions provided by the manufacturer or your instructor. Additionally, always make sure you are properly trained and familiar with the spin recovery procedure before attempting it.

27.01.2023

What changed? Presumably, enough people complained and the system was re-trained with correct responses.

# Vision Transformers



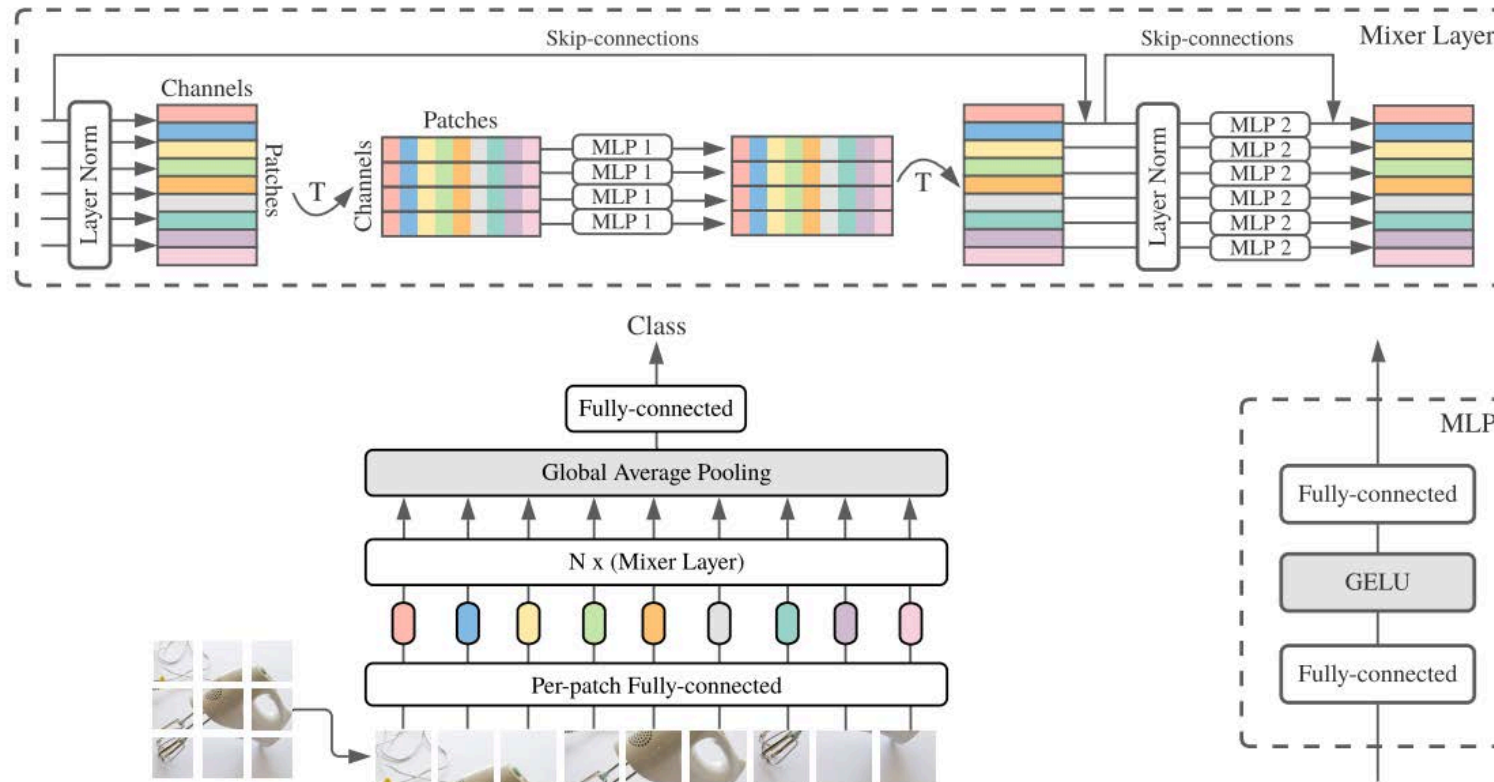
$$\mathbf{X}_1 = \Phi(\mathbf{P}_1)$$

$$\mathbf{X}_2 = \Phi(\mathbf{P}_2)$$

$$\mathbf{X}_3 = \Phi(\mathbf{P}_3)$$

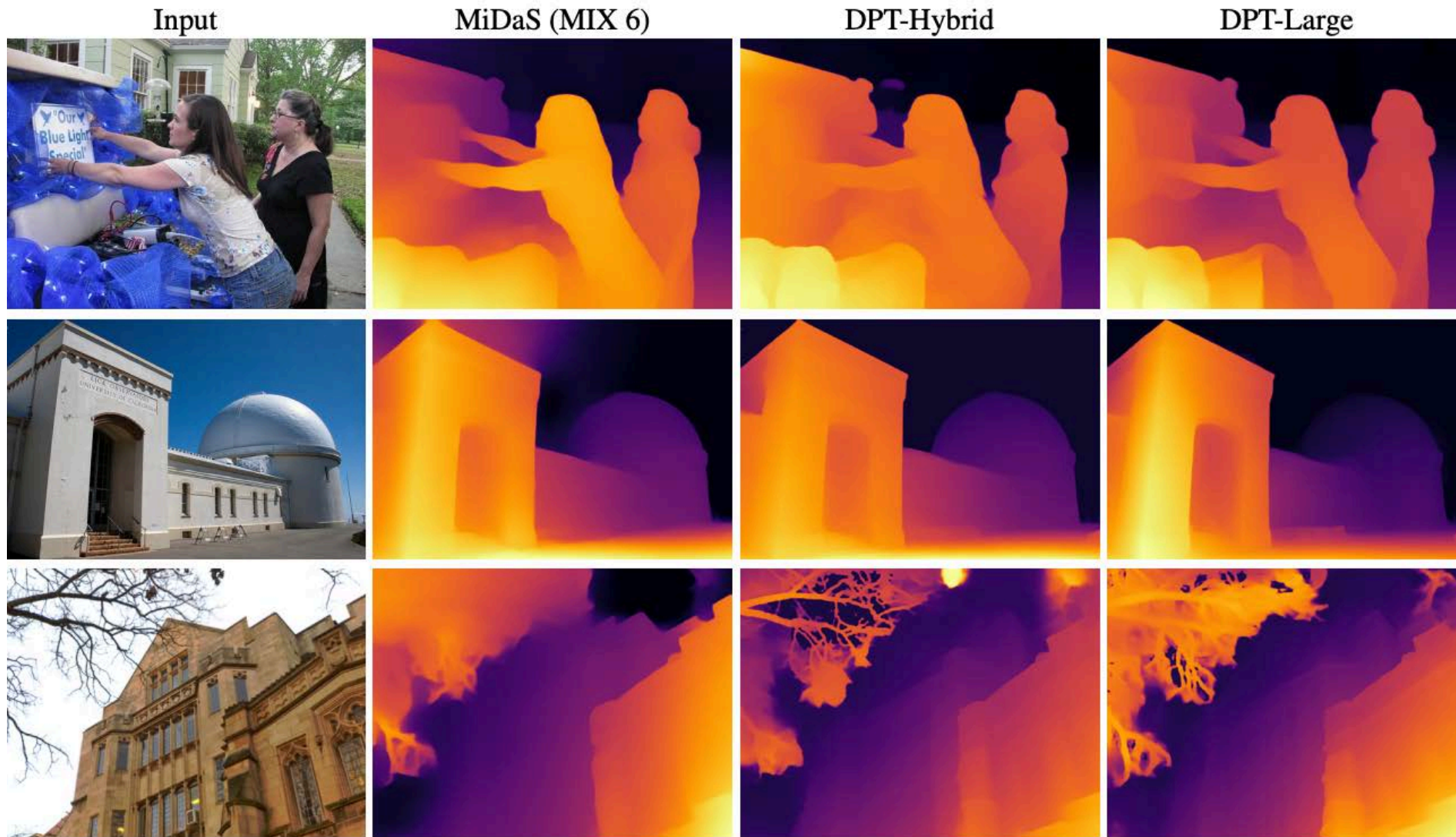
$$\mathbf{X}_4 = \Phi(\mathbf{P}_4)$$

# MLP Mixers



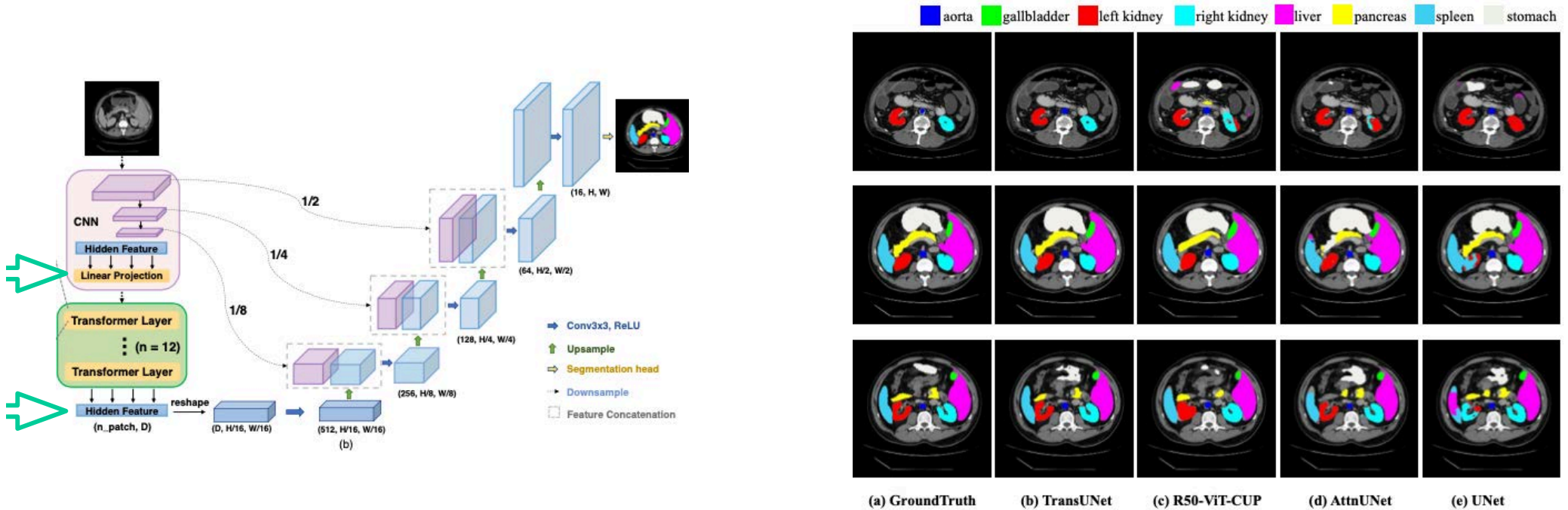
- Break up the images into square patches.
- Transform each path into a feature vector.
- Feed to a transformer architecture.

# Depth from Single Images



- Pros: Good at modeling long range relationships.
- Cons: Flattening the patches loses some amount of information.

# U-NET + Transformers



- A CNN operates at low-resolution and produces a feature vector.
- A transform operates on that feature vector.
- The upsampling is similar to that of U-Net

—> Best of both worlds?