

10417/10617

Neural Network for Vision Part III

Vision Language Models

- One of the most impressive achievements recently is vision language models. (Generation from phi-1.5-vision, a 1.3B parameter vision language model)

What is in this image <|img_1|>?

Answer: The image features a group of men sitting in a military-style vehicle, which is a tank. The men are dressed in military uniforms, indicating that they are part of a military unit or group. The tank is parked in a field, surrounded by trees and other natural elements.

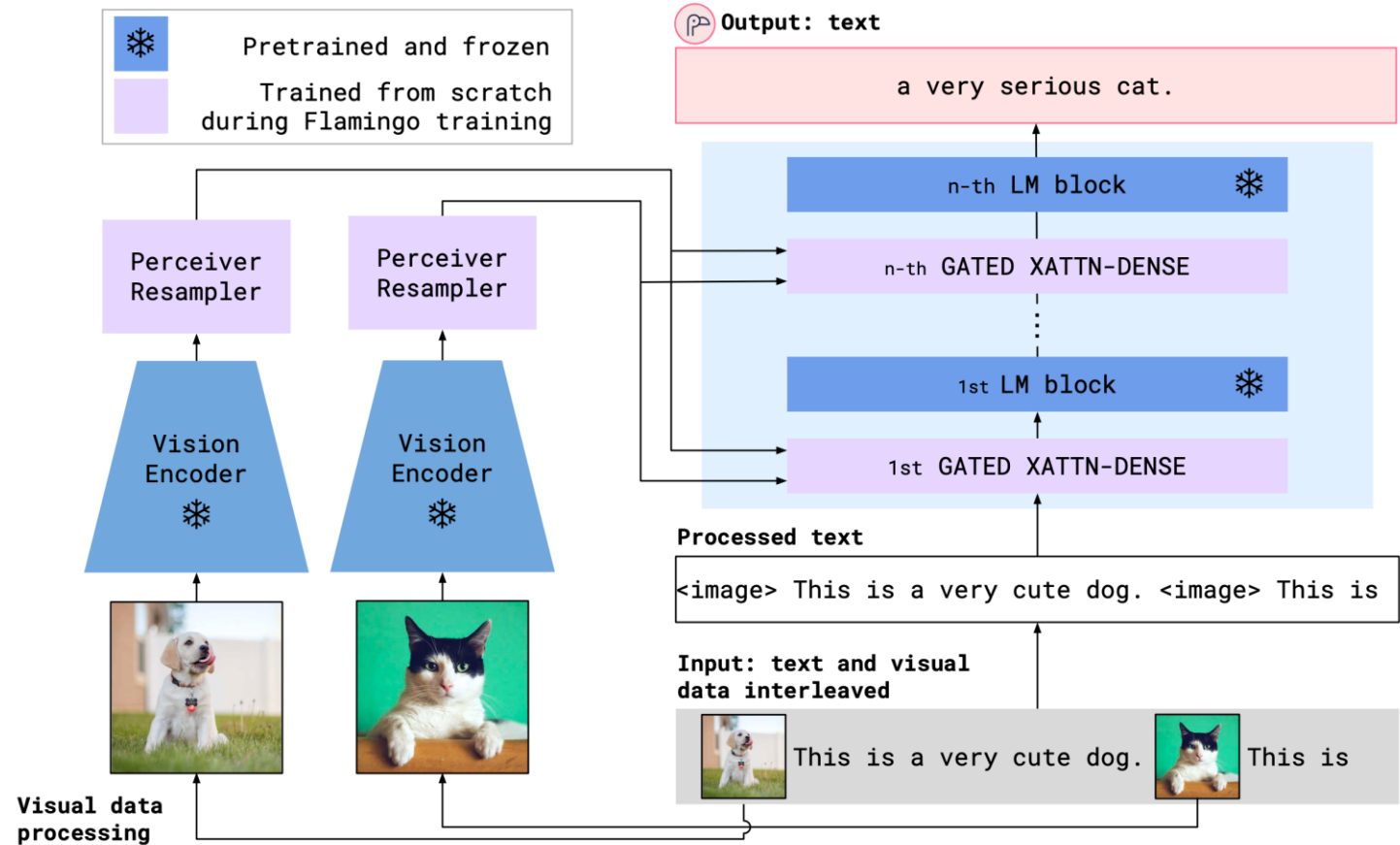
The group of men is likely engaged in some sort of military activity, such as training, exercise, or possibly a mission. The presence of the tank suggests that they are part of a military unit or group, which could be a tank regiment or a special forces unit.

<|endofplaintext|>



So how do we get such a model?

- There are three parts:
 - Vision encoding part → We will partially cover today.
 - Language modeling part → We will partially cover today (transformer models).
 - Vision and language connection part → We leave for the future (multi-modality).



Vision Encoder

A vision encoder is a network that maps a given image to a vector.

The vector “encodes” the image in a more “decodable” way.

For example, we can apply a linear function over the vector to classify its labels.

- You can think of one “encoder” as the last hidden layer in a convolution network.

For example, we can apply some simple functions to the encoding to perform objects segmentation.

Vision Encoder

- How do we train a vision encoder?
 - One way: We can train a convolution neural network for image classification, and take its last hidden layer.
 - Disadvantage: The vision encoder is only good for images **similar to the training image**, not good for some general images on the internet.

What is in this image <|img_1|>?

In this image, a banana-shaped vehicle is parked on the road, which is a unique and creative concept. The vehicle is a modified car, with a banana-shaped body and wheels. It is parked on a road, which is a common sight in many cities. The banana-shaped vehicle is a fun and quirky addition to the urban landscape, and it is likely to attract attention and curiosity from passersby.

<|endoftext|>



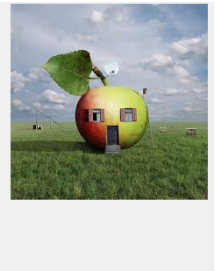
Vision Encoder

- How do we get a vision encoder that is good for all (or most of) the images on the internet?
- We do not have labels for a (vast) majority of those images (we can't even label them).
- We need to train a good “image encoder” without any labels on those images.
 - How??? We have no idea about those images, and we want to train a good encoder to encode good ideas (good features) about those images...
 - Well, it is doable (like magic). Although the method is still not really well understood even today...

What is in this image `<|img_1|>`?

In this image, a large, green, apple-shaped house is located in a field with a house-like structure. The house is surrounded by a variety of objects, including small, green, house-like structures.

`<|endoftext|>`



Contrastive Learning for Vision Encoder

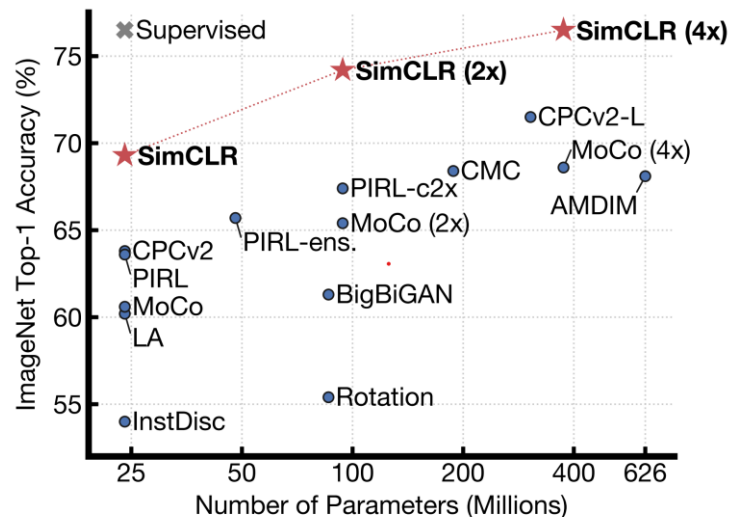
- The idea is to apply contrastive learning here. The famous method is called SimCLR (Hinton et al. 2021).
- Given a vast amount of training images $x^{(1)}, x^{(2)}, \dots, x^{(n)}$ (without labels), the contrastive loss is defined as:
- $$\min_h \frac{1}{N} \sum_i - \log \frac{\exp\{\langle h(x^{(i)}), h(\text{aug}(x^{(i)})) \rangle / \tau\}}{\frac{1}{m} \sum_{j \in S_m} \exp\{\langle h(x^{(i)}), h(x^{(j)}) \rangle / \tau\}}$$
- Here, τ is the temperature, and $\text{aug}(x)$ means the augmentation of x .
- S_m is a random sample of batches of m images from $x^{(1)}, x^{(2)}, \dots, x^{(n)}$ (without $x^{(i)}$).
- h is usually a convolution neural network.
 - The output of h is a vector, and it is layer-normalized (so it has norm one).

Contrastive Loss

- $\frac{1}{N} \sum_i -\log \frac{\exp\{\langle h(x^{(i)}), h(\text{aug}(x^{(i)})) \rangle / \tau\}}{\frac{1}{m} \sum_{j \in S_m} \exp\{\langle h(x^{(i)}), h(x^{(j)}) \rangle / \tau\}}$
- To minimize this loss, we want:
 - $h(x^{(i)})$ is close to $h(\text{aug}(x^{(i)}))$, so the output of the vision encoder is robust under data augmentation.
 - $\exp\{\langle h(x^{(i)}), h(x^{(j)}) \rangle / \tau\}$ is small, so $h(x^{(i)})$ should (not be correlated)/(anti correlated) with $h(x^{(j)})$ for different images.
- So we just need a diversified set of features robust to data augmentation... and that's it.

The power of contrastive learned features

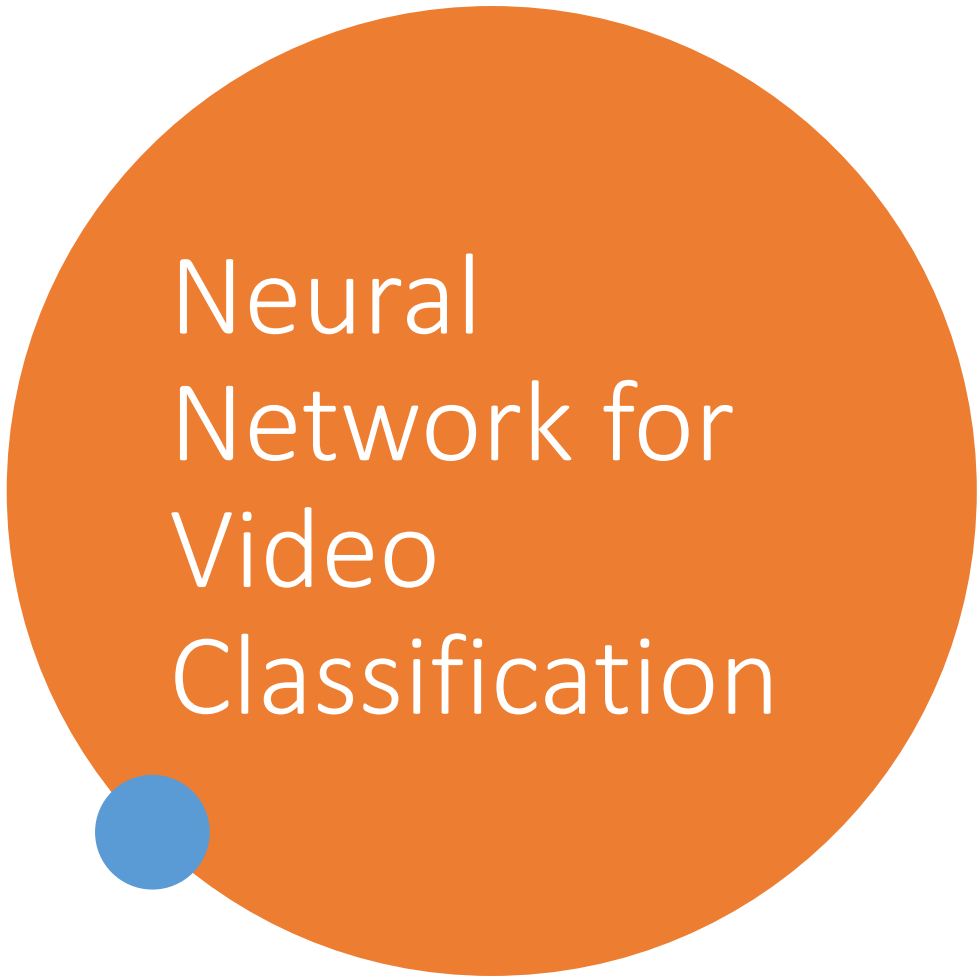
- After getting contrastive learned features $h(x)$, how do we test whether it is good?
- One way is to **train a linear function** on top of it, and see how does $f(x) = \langle w, h(x) \rangle$ perform to classify the labels of the images.



Baseline: Linear function over randomly initialized h
Less than 10% accuracy.

Neural Network for Video Classification

- Now we learned image classification, can we actually use neural networks for video classification?
- Video can be viewed as a 4d tensor, $(d \times d \times C) \times T$
- There is an additional time dimension T .



Neural Network for Video Classification

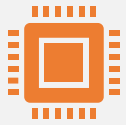
- There are two different approaches to solve this:
 - Using a standard convolution network + LSTM/Transformer to process sequential input (will be covered in later lectures).
 - Using a 3D convolution network.



3D convolution.

- Given an input $x \in (d \times d \times C) \times T$, a 3D convolution operation is defined as:
- $y[i, j, k, r] = \sum_{r, s \in [p], t \in [P], l \in [C]} w[r, s, k, l, t] x[m \times i + r, m \times j + s, l, m \times r + t] + b[k]$
- W is of size $p \times p \times C' \times C \times P$
- So it is applying convolutions on 3D: $d \times d \times T$.

Neural Network for Vision From Convolution to Vision- Transformer



Is the convolution
network still used
today?

Not at all in most of
the frontier
applications...

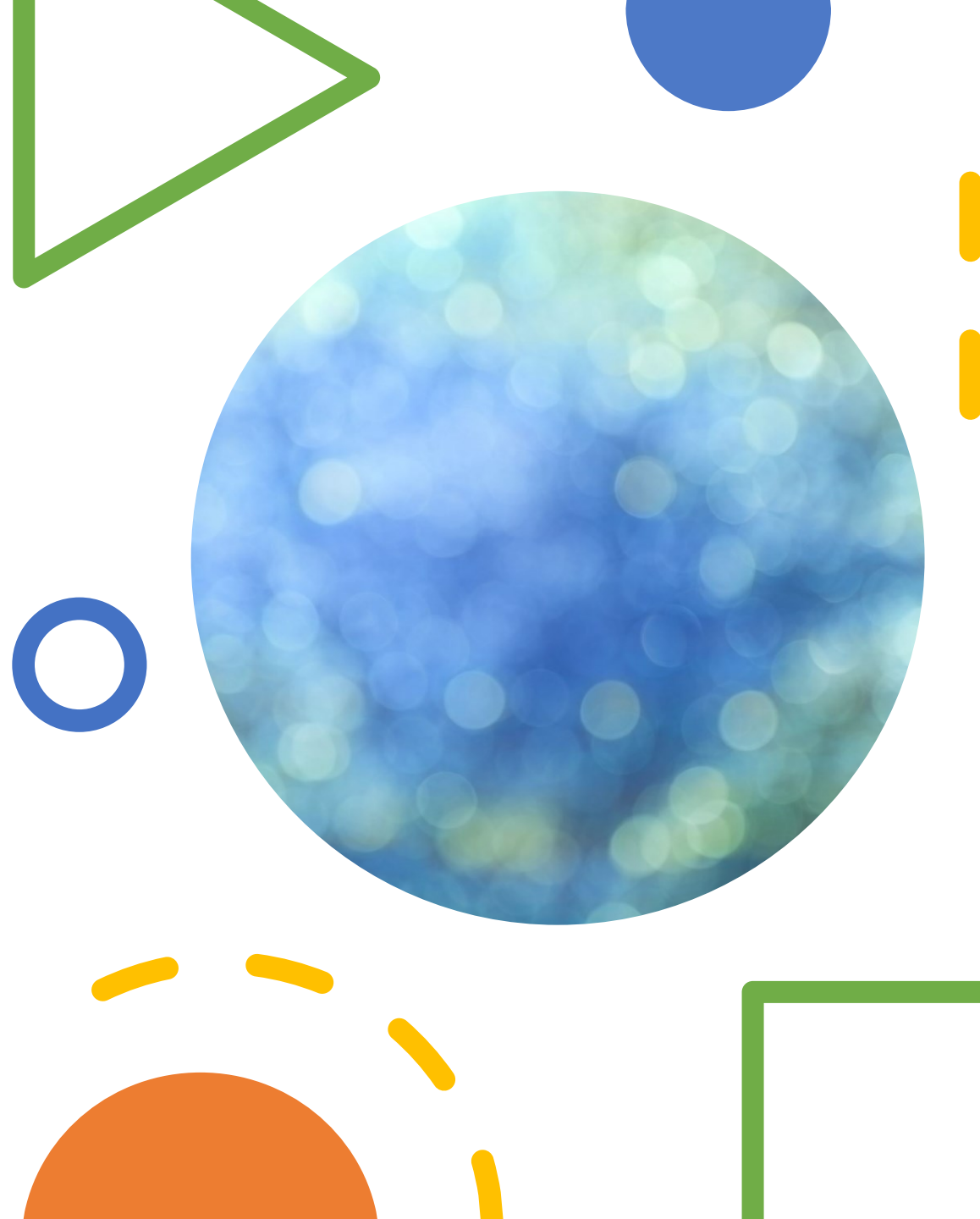
It is completely
replaced by Vision
Transformer (ViTs)
that was introduced
in 2021.

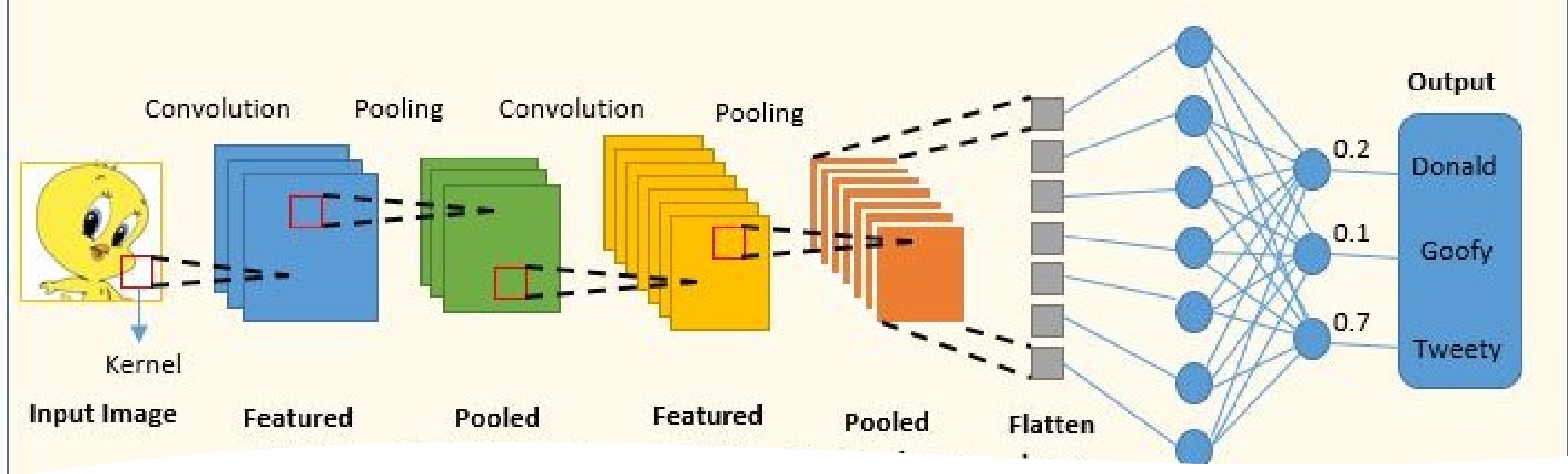


Why?



成也convolution,败也
convolution...



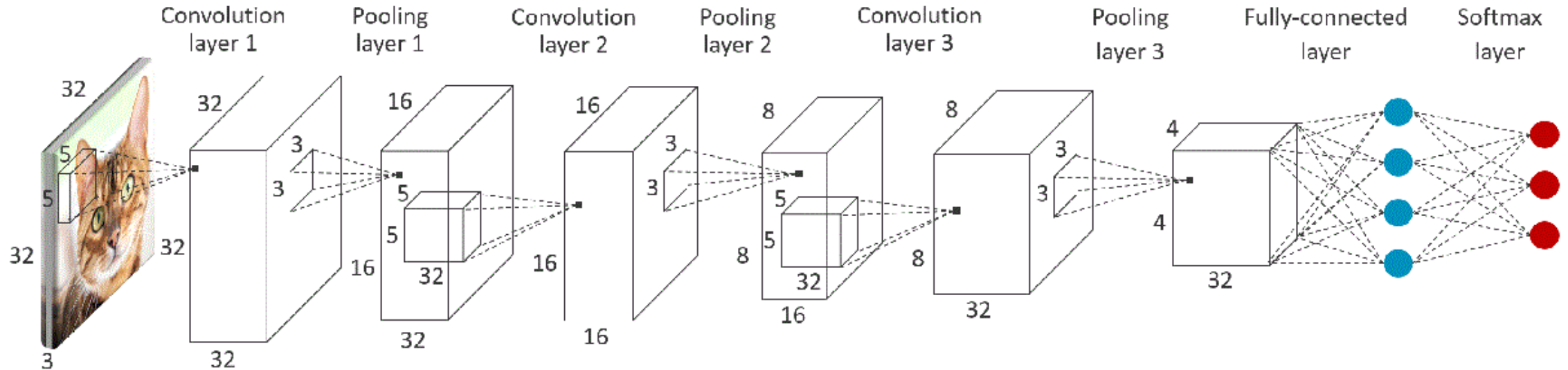


Neural Network for Vision From Convolution to Vision- Transformer

- 成也convolution,败也convolution...
- One good side that allows convolution to outperform MLP is also the downside that lets convolution losses to transformer completely...
- Good side about convolution: Convolution is a local operation, so its parameter count is $\text{kernel_size} \times \text{kernel_size} \times \text{in_channel} \times \text{out_channel}$ (regardless of the input size).

From Convolution to Vision-Transformer

- Good side about convolution: Convolution is a local operation, so its parameter count is $\text{kernel_size} \times \text{kernel_size} \times \text{in_channel} \times \text{out_channel}$ (regardless of the input size).
 - So it is more sample/computation efficient.
- Bad side: However, convolution is a local operation, so it can not capture the global information of the image very well.



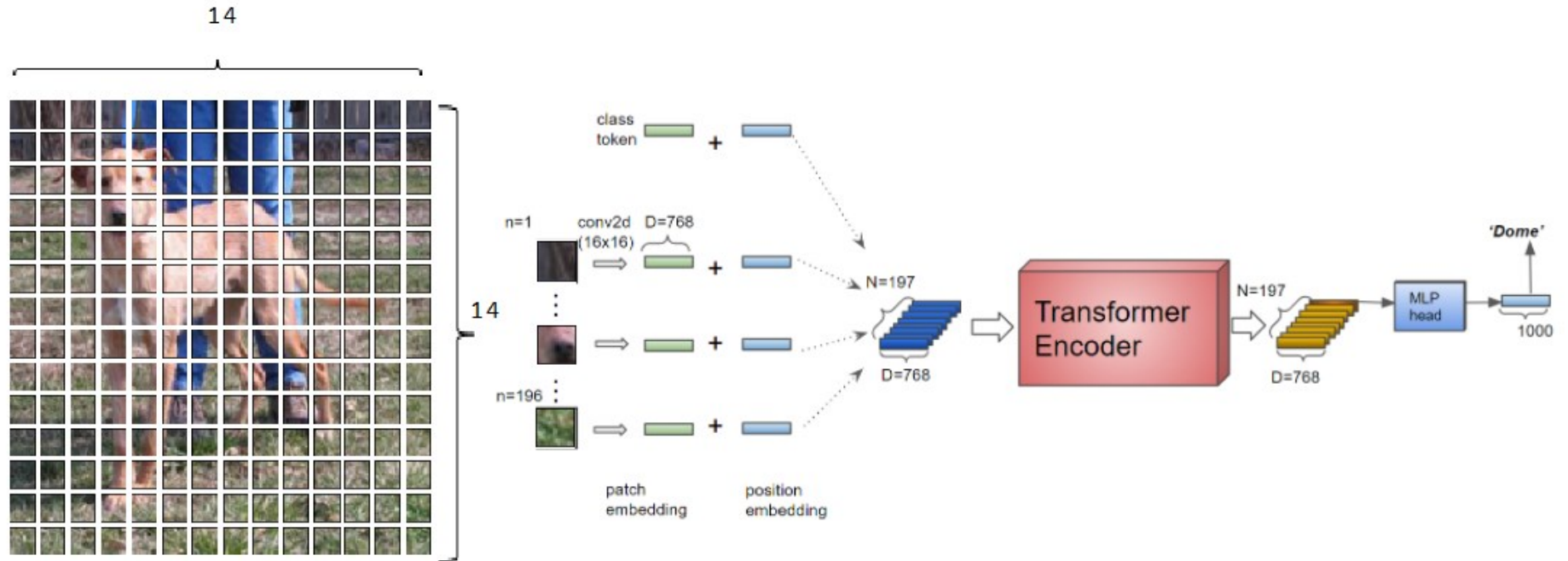
From Convolution to Vision-Transformer

- Far away correlations in the image might get lost in a convolution network.

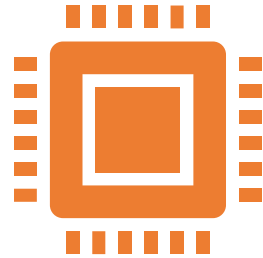


Vision-Transformer

- Question: Can we design an architecture that does local operations, but still keeps the global information?
- This is the intuition behind vision transformer.



Vision-Transformer



Idea behind Vision Transformer

We would like to apply the same local operations on the patches of the image (to be computation efficient and reduce the number of parameters).

We would like to have some “global operations” that mix features from far away patches.



Such an architecture is called a **transformer**.



Vision Transformer

- Basic structure of ViT:
 - Step 1, divide the input images ($d \times d \times C$) into (d^2/p^2) (disjoint) patches, each of size $(p \times p \times C)$.
 - Step 2:
 - Step 2.1. For each patch, flatten it, apply an MLP on it, and get an output of size d_{emb}
 - Step 2.2. For each vector on each patch, “mix” them together, to create new vectors on each patch.
 - Repeat Step 2.
-

Vision-Transformer



- Step 2.1 is easy, but what is step 2.2?
- The mixing operation is called “self-attention”.
- Given vectors v_1, v_2, \dots, v_n on n patches, each of dimension D , a self-attention operator returns vectors v'_1, v'_2, \dots, v'_n , each of dimension D .
- $v'_i = \sum_j \alpha_{i,j} v_j$ as a weighted average of the input vectors v 's.

Vision Transformer:

- Input image of size $(d \times d \times C)$
- -> Divided into (d^2/p^2) patches, flatten each patch to be a vector in $p^2 C$.
- -> Linear layer on each patch (to increase the dimension to d_{emb}).
- (-> Self-attention on all the vectors -> apply the same one-hidden-layer-MLP per layer on each and every vector) $\times L$
- -> flatten -> one-hidden-layer MLP
- This is called a L-layer vision transformer.