Introduction to Machine Learning III

Harrison B. Prosper

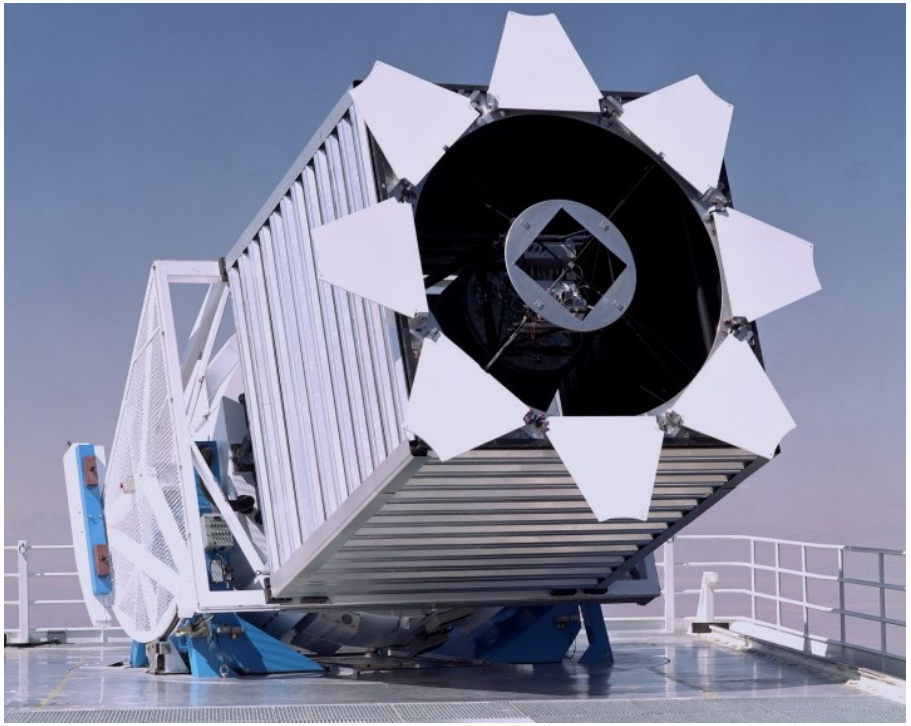
Florida State University

Topics

- Auto-Encoders (AE)
- Convolutional Neural Networks (CNN)
- Transformer Neural Networks (TNN)
- Summary

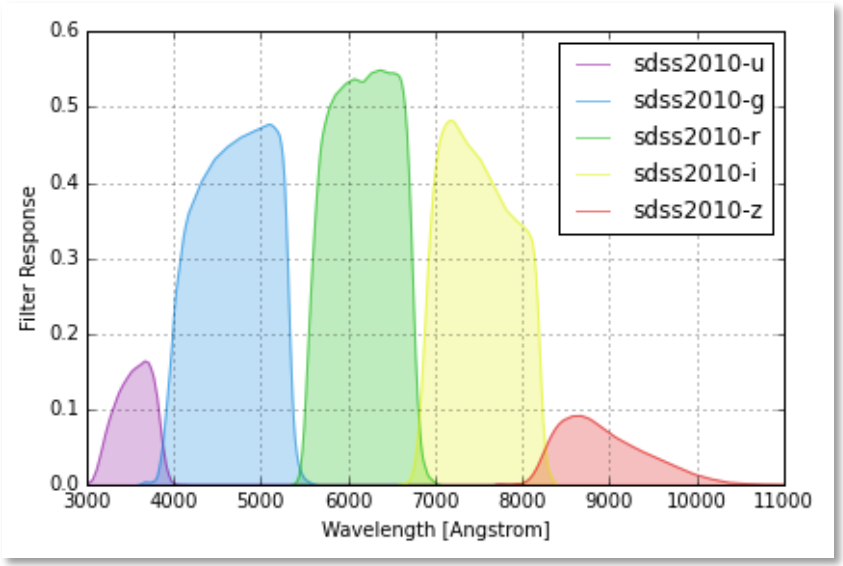
Sloan Digital Sky Survey (SDSS)

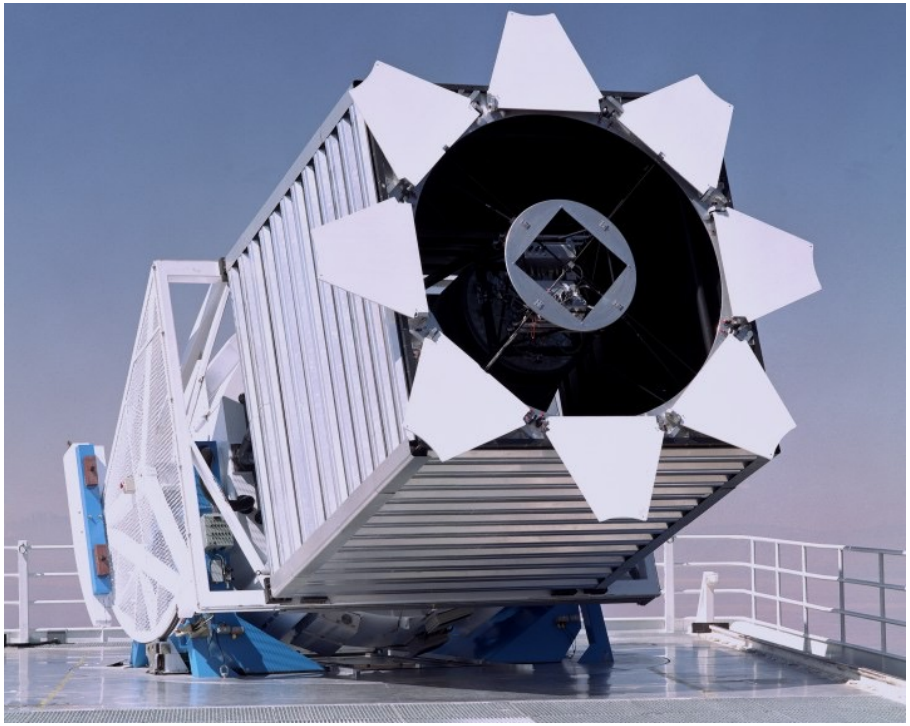
The goal of the SDSS, which began twenty years ago, is to create a 3D map of the universe and make these data publicly available for scientific and educational purposes.



The main instrument is the 2.5m telescope at Apache Point Observatory in New Mexico, USA.

Sloan Digital Sky Survey (SDSS)





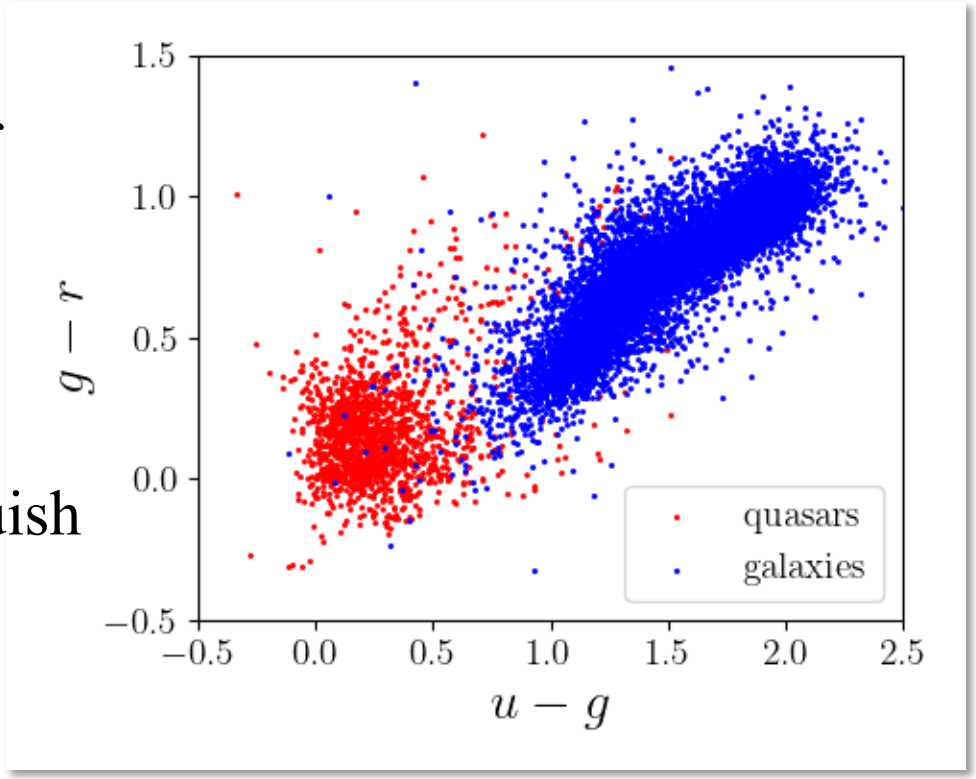
The sky is observed through Several filters labeled **u** (ultraviolet), **g** (green), **r** (red), **i** (near infrared), and **z** (infrared).

https://speclite.readthedocs.io/en/latest/filters.html

https://www.sdss.org/instruments/camera

Sloan Digital Sky Survey (SDSS)

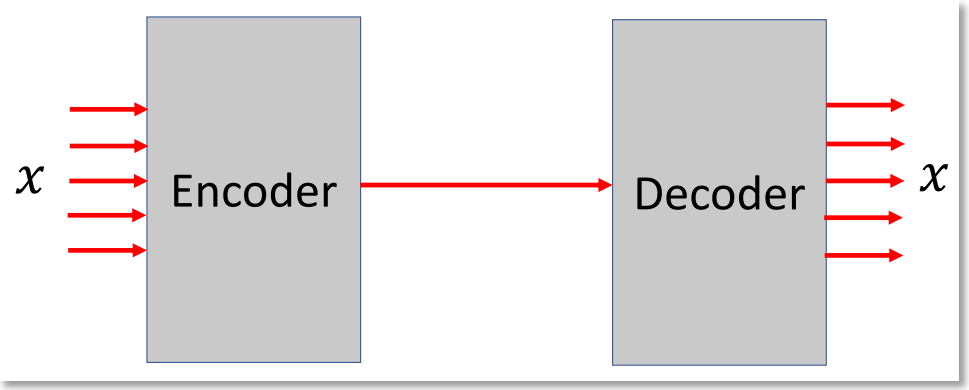
Astronomers have known for a long time that a plot of $g - r$ vs. $u - g$ shows an excellent clustering of quasars and galaxies. This makes it possible to distinguish quasars from galaxies.



Question: could such a clustering have been discovered using automated means? Yes, for example, using an auto-encoder.

Auto-Encoder (1)

An auto-encoder (AE) is a neural network that implements the mappings $h: x \rightarrow z$ and $g: z \rightarrow x$, where $x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$ with $m \ll n$.



Here is an example with $n = 5$ and $m = 1$. The space $\mathbb{R}^m$ is called the latent space.

If there is structure within the data, one would expect some of that structure to be preserved within the latent space.

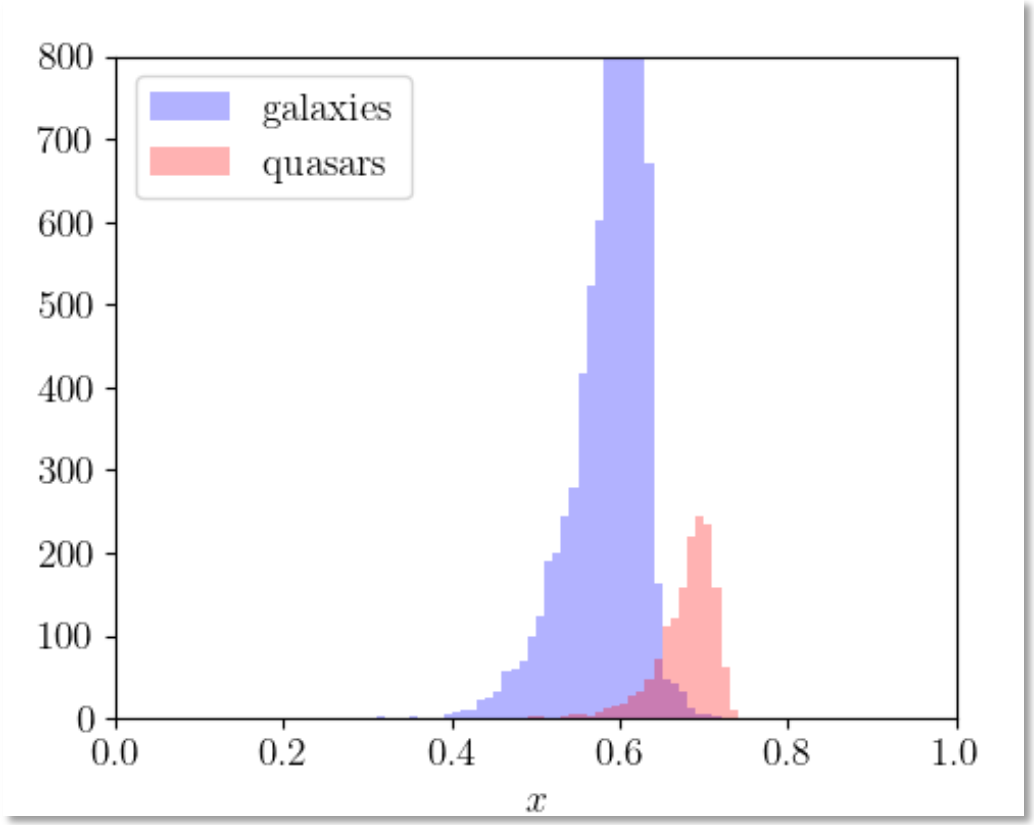
Auto-Encoder (2)

A simple auto-encoder that maps a 5D space to a 1D latent space.

```
class AutoEncoder(nn.Module):

    def __init__(self):
        """
    A simple fully-connected autoencoder neural network.
        """
        # call constructor of base class
        super(AutoEncoder, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(5 ,  20), nn.ReLU(),
            nn.Linear(20,  20), nn.ReLU(),
            nn.Linear(20,   1), nn.Sigmoid())

        self.decoder = nn.Sequential(
            nn.Linear(1,   20), nn.ReLU(),
            nn.Linear(20,  20), nn.ReLU(),
            nn.Linear(20,  5))

    def forward(self, x):
        y      = self.encoder(x)
        self.z = y # cache latent space
        y      = self.decoder(y)
        return y
```

Auto-Encoder (3)

The auto-encoder finds that there are indeed two classes of objects in the SDSS color data.

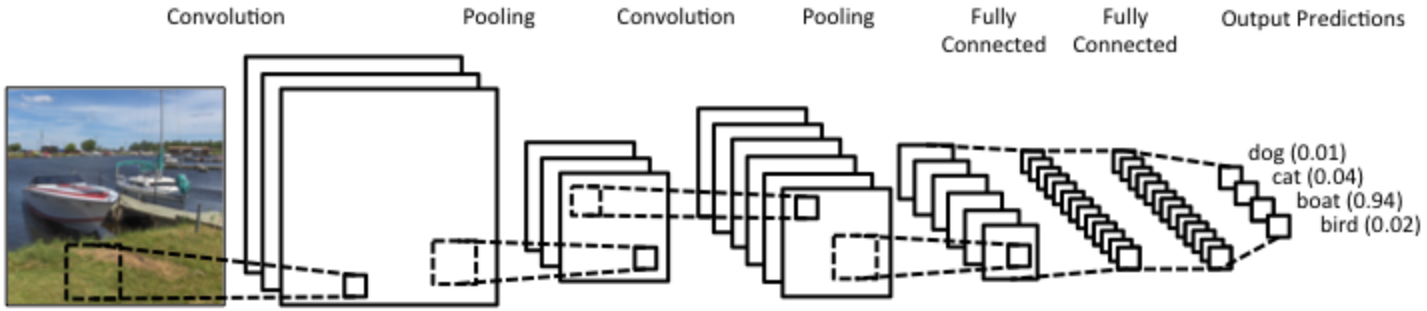


CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (1)

Many of the breakthroughs in tasks such as face recognition use a type of deep neural network (DNN) called a convolutional neural network (CNN).

CNNs are *functions* that compress data and classify objects based on their compressed representations using a fully connected NN. The compression dramatically reduces the dimensionality of the space to be searched.



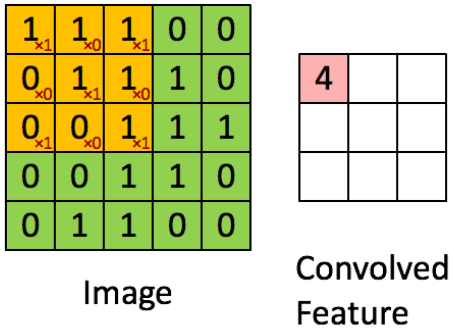
Source: https://www.clarifai.com/technology

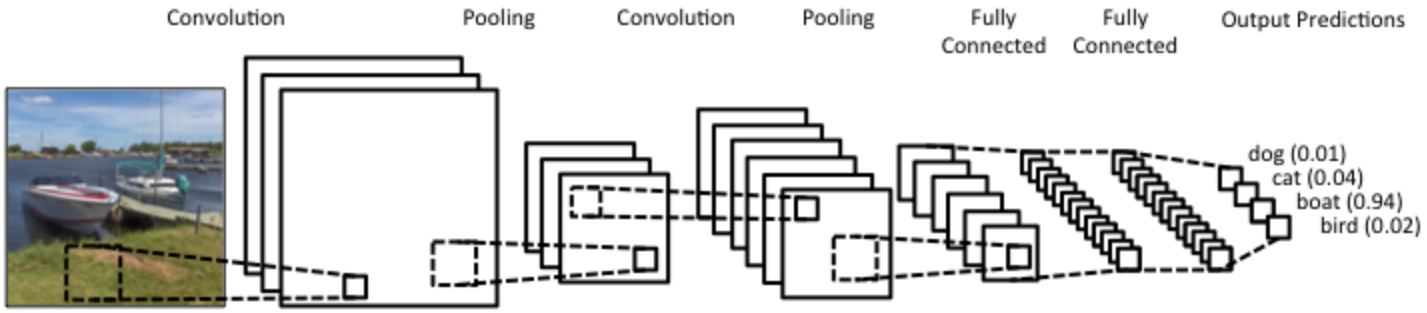
Convolutional Neural Networks (2)

A CNN comprises three types of processing layers:
1. convolution, 2. pooling, and 3. classification.

1. **Convolution layers**

   The input layer is "convolved" with one or more matrices using element-wise products that are then summed. In this example, since the sliding matrix fits 9 times, we compress the input from a 5 x 5 to a to a 3 x 3 matrix.

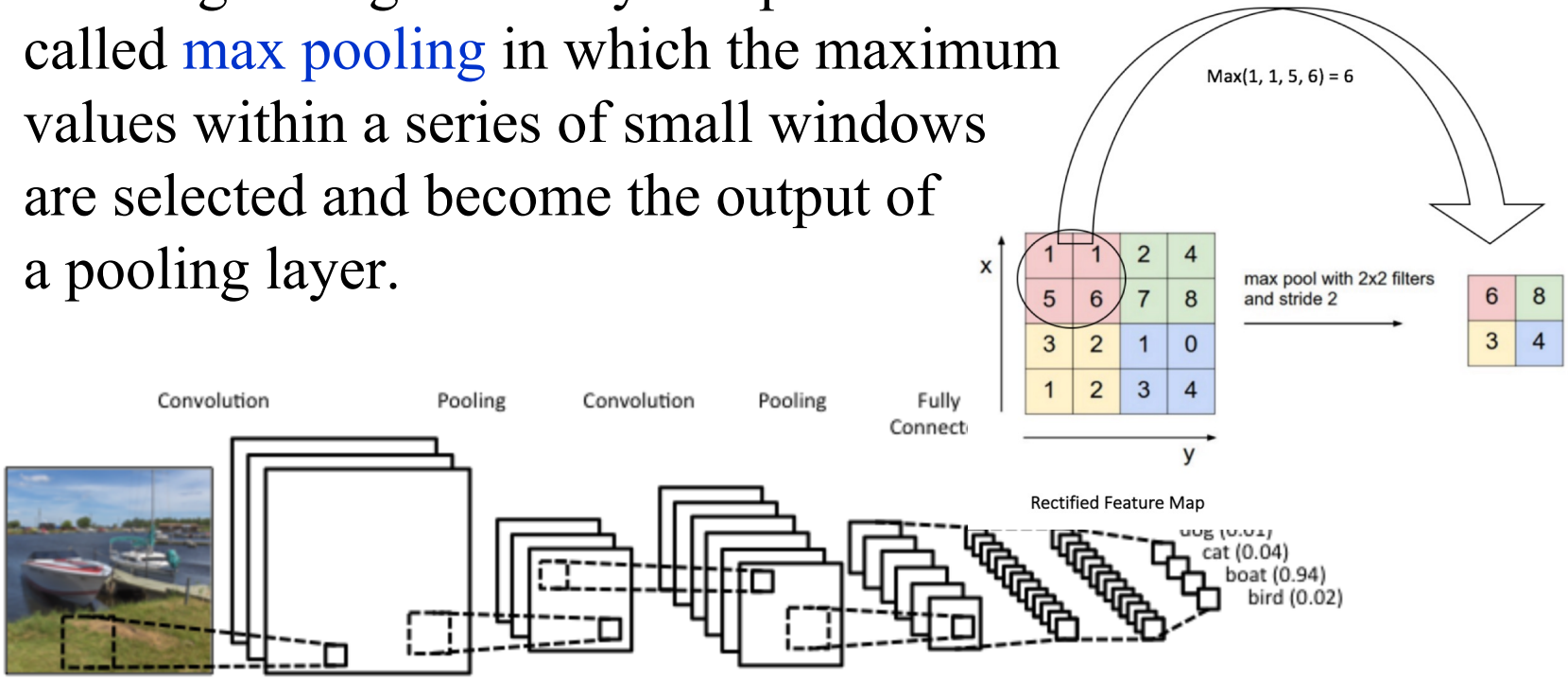




Convolutional Neural Networks (3)

2. Pooling Layers

After convolution, and a pixel-by-pixel non-linear map (using, e.g., the function $y = \max(0, x) = \text{ReLU}(x)$), a coarse-graining of the layer is performed called max pooling in which the maximum values within a series of small windows are selected and become the output of a pooling layer.

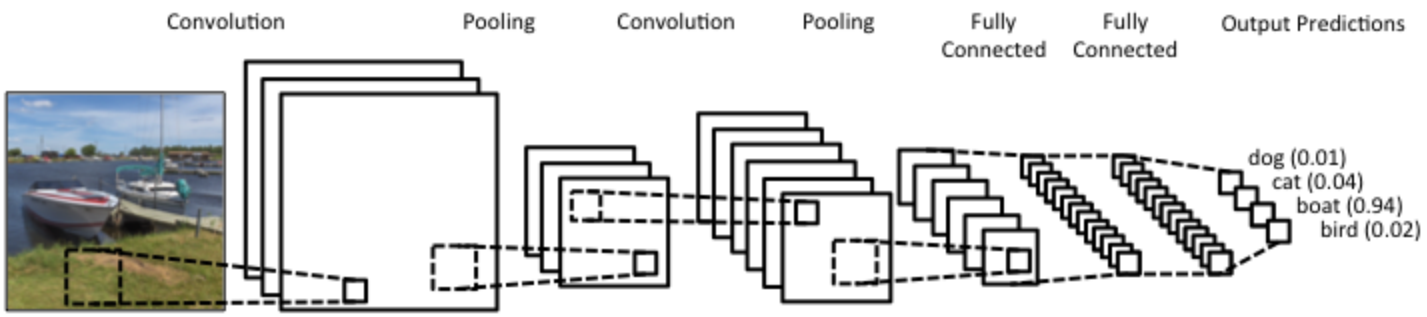


Convolutional Neural Networks (4)

3. Classification Layers

After an alternating sequence of convolution and pooling layers, the outputs go to a standard neural network, either shallow or deep. The final outputs correspond to the different classes and like all flexible classifiers, a CNN approximates,

$$p(C_k|x) = p(x|C_k)p(C_k) / \sum_{m=1}^{M} p(x|C_m)p(C_m)$$



TRANSFORMER NEURAL NETWORKS

Transformers (1)

A transformer[1] is a function that takes *in parallel* an entire sequence of source tokens (e.g., of English)

$$S = [<sos>, x_1, \cdots, x_n, <eos>]$$

together with a start-of-sequence target token <sos> and computes for the *next* output token a discrete probability distribution

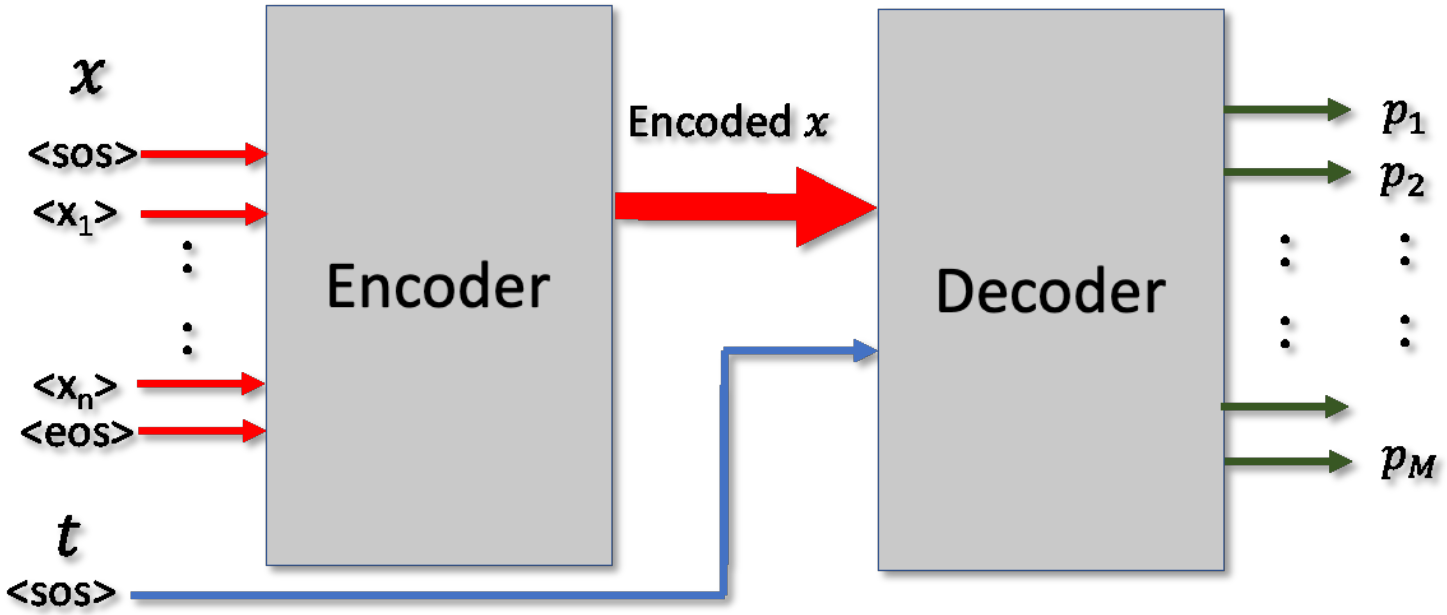
$$P = p_1, \cdots, p_M$$

over a target vocabulary of size M (e.g., of German tokens).

1. https://arxiv.org/abs/1706.03762

Transformers (2)

The model is autoregressive:



Transformers (4)

Example: suppose we want to build function that can expand a certain class of functions as a Taylor series to 6th order:

$$x \quad \left(e^{3ax} - \frac{\sin(cx)}{\sinh(gx)}\right) e^{-ax}$$

$$1 + x\left(\frac{ac}{g} + 2a\right) + x^2\left(-\frac{a^2c}{2g} + 2a^2 + \frac{c^3}{6g} + \frac{cg}{6}\right)$$

$$t \quad + x^3\left(\frac{a^3c}{6g} + \frac{4a^3}{3} - \frac{ac^3}{6g} - \frac{acg}{6}\right)$$

$$+ x^4\left(-\frac{a^4c}{24g} + \frac{2a^4}{3} + \frac{a^2c^3}{12g} + \frac{a^2cg}{12} - \frac{c^5}{120g} - \frac{c^3g}{36} - \frac{7cg^3}{360}\right)$$

$$+ x^5\left(\frac{a^5c}{120g} + \frac{4a^5}{15} - \frac{a^3c^3}{36g} - \frac{a^3cg}{36} + \frac{ac^5}{120g} + \frac{ac^3g}{36} + \frac{7acg^3}{360}\right) - \frac{c}{g} + O\left(x^6\right)$$

Transformers (5)

1. Build the vocabularies for the source and target sequences and code each token as an integer,

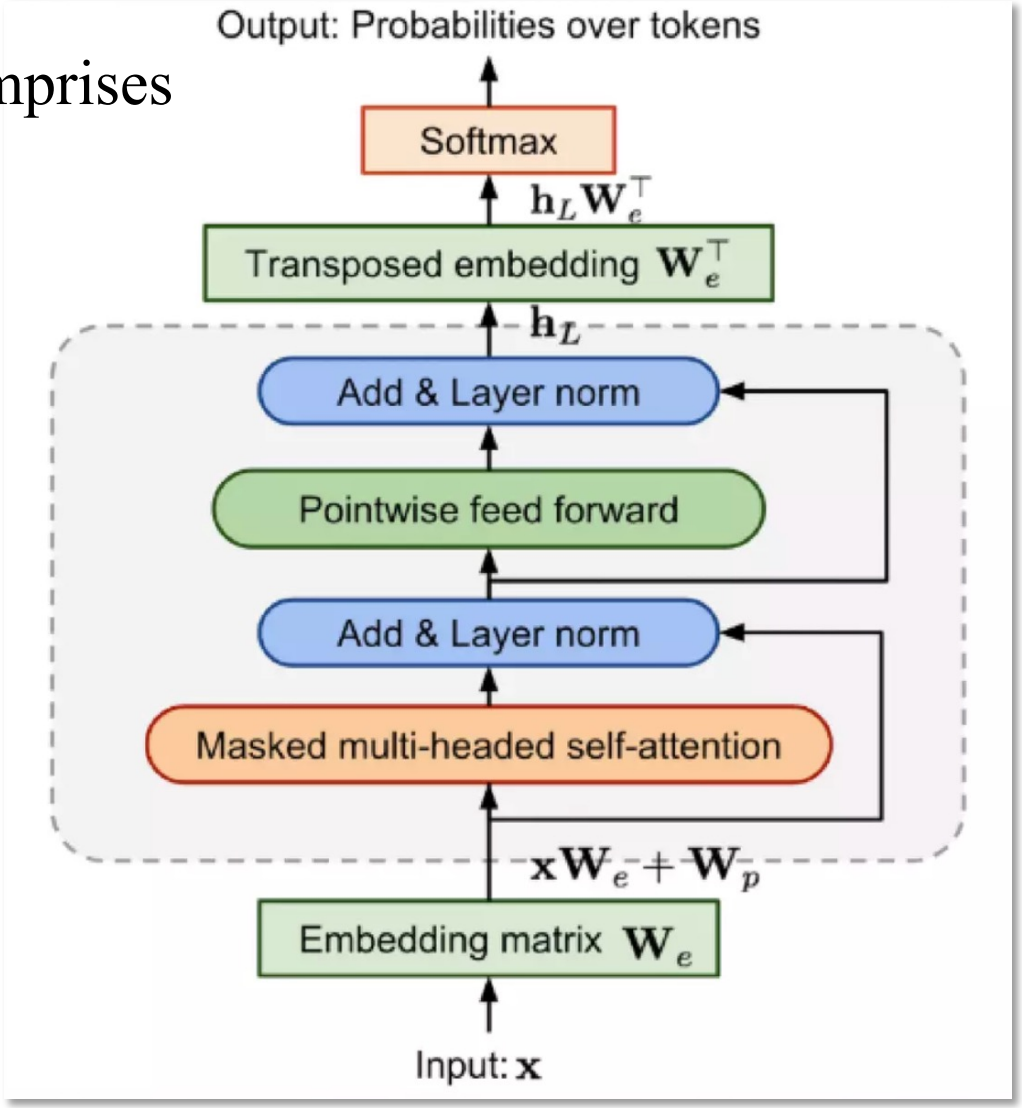
```
source vocabulary
{'<pad>': 0, '<sos>': 1, '<eos>': 2, '(': 3, ')': 4, '*': 5, '**': 6,
'+': 7, '-': 8, '/': 9, '0': 10, '1': 11, '2': 12, '3': 13, '4': 14, '5
': 15, '6': 16, '7': 17, '8': 18, '9': 19, 'a': 20, 'b': 21, 'c': 22, '
cos': 23, 'cosh': 24, 'd': 25, 'exp': 26, 'f': 27, 'g': 28, 'sin': 29,
'sinh': 30, 'tan': 31, 'tanh': 32, 'x': 33}

target vocabulary
{'<pad>': 0, '<sos>': 1, '<eos>': 2, '(': 3, ')': 4, '*': 5, '**': 6,
'+': 7, '-': 8, '/': 9, '0': 10, '1': 11, '2': 12, '3': 13, '4': 14, '5
': 15, '6': 16, '7': 17, '8': 18, '9': 19, '0(x**6)': 20, 'a': 21, 'b':
22, 'c': 23, 'd': 24, 'f': 25, 'g': 26, 'x': 27}
```

2. Using the vocabularies, *tokenize* every sequence and map the tokens to integers.

Transformers (6)

The transformer model comprises three computational layers:

1. Embedding
2. Multi-Head Attention
3. Probability Calculator



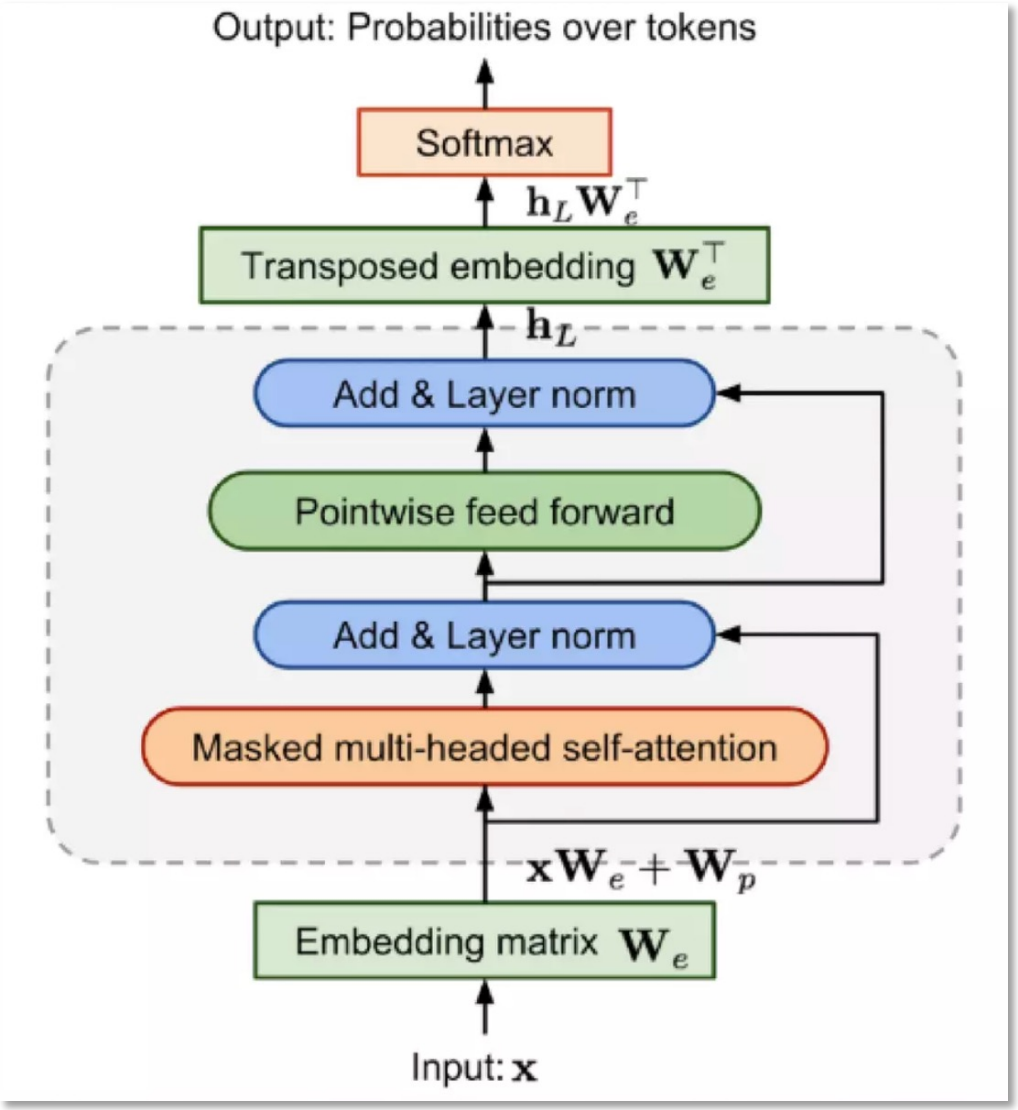
Credit: Steve Omohundro / OpenAI
https://steveomohundro.com/

Transformers (7)

1. Embedding

Map every token in

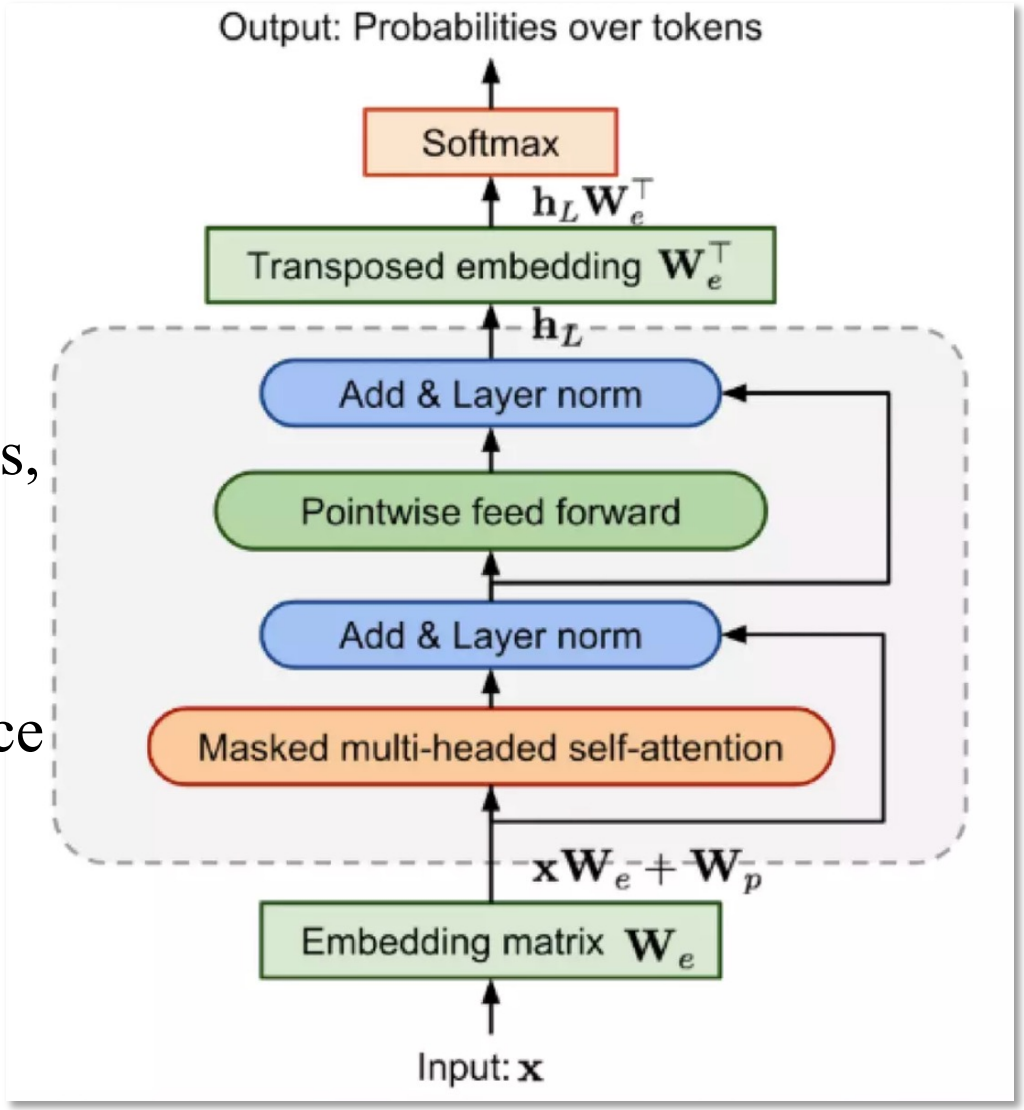
a sequence a large vector.



Transformers (8)

2. Multi-Head Attention

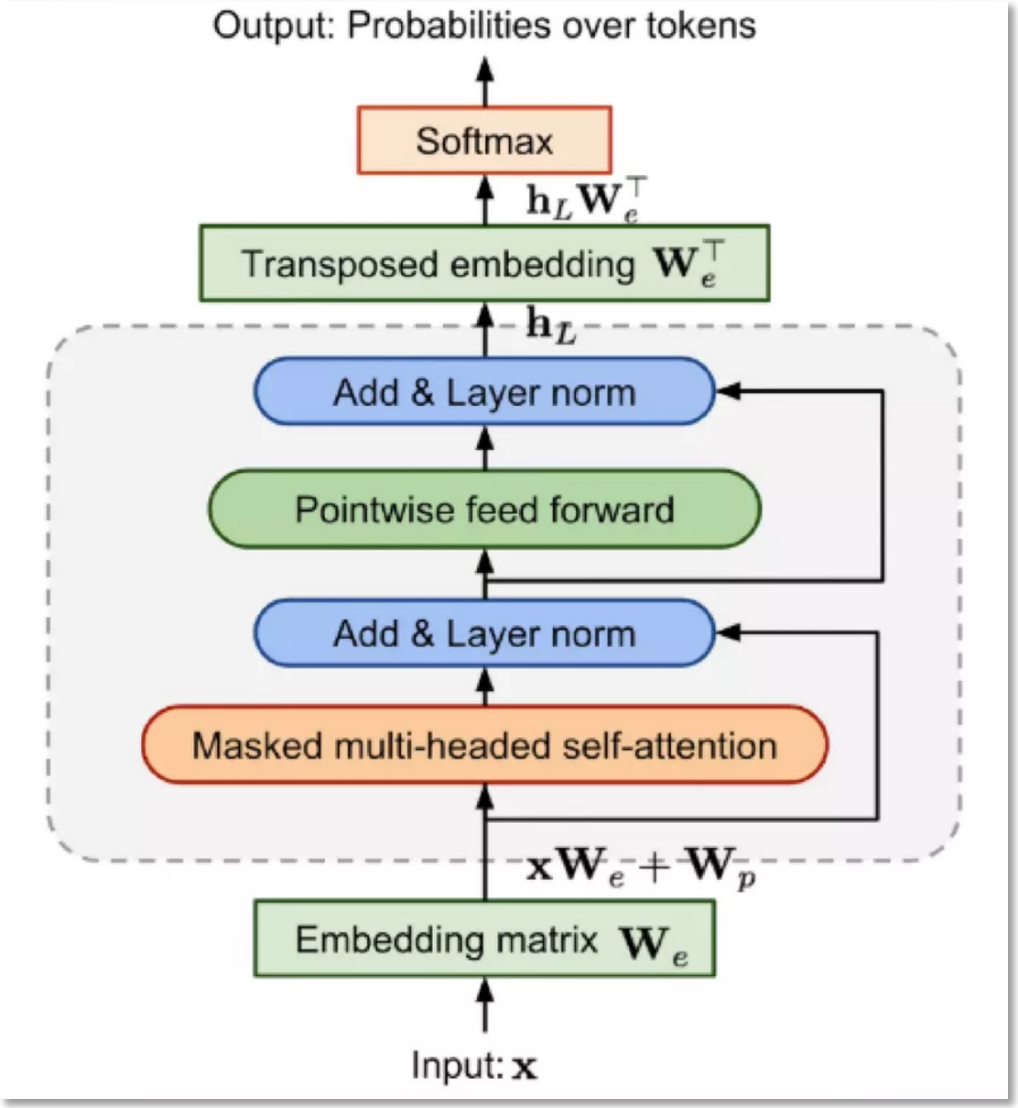
Transform each token vector into another that encodes the strength of the relationship between tokens, both within the same sequence, and within the decoder also between source and target sequences.



Transformers (8)

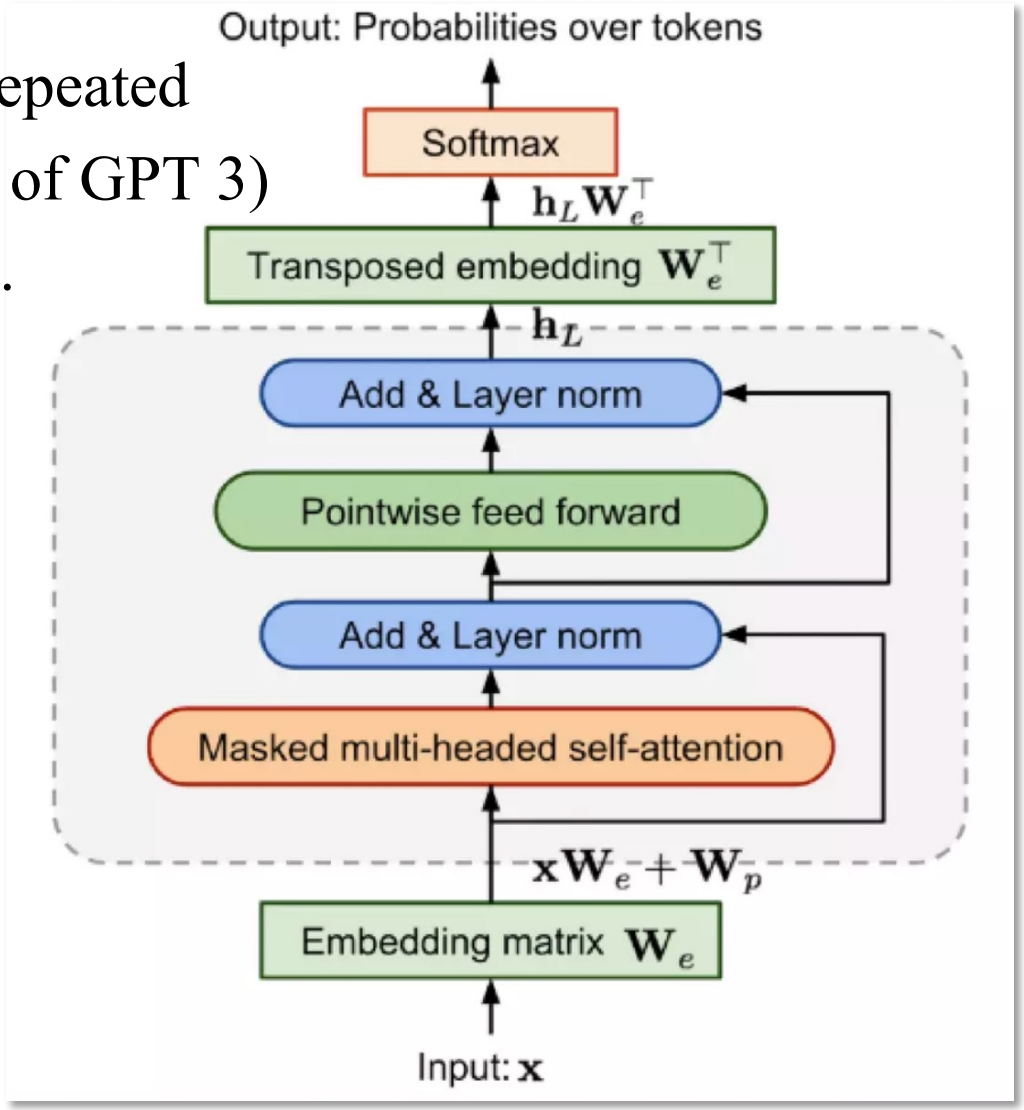
3. Probability Calculator

Compute a discrete probability distribution over target vocabulary.



Transformers (9)

The grey block is usually repeated many times (96 in the case of GPT 3) in the encoder and decoder.



Summary

- Auto-encoders are a useful model for creating compressed representations of data. By constraining the latent space with an additional loss term extra structure can be induced in that space.

- Transformers are complicated functions, albeit built from very simple computational units. These functions contain several features that seem *ad hoc*.

- But transformers are now the state-of-the-art in ML/AI.