Convolutional Neural Networks

Topics covered so far

- Convolutional Neural Networks
  - Locality, Translation invariance
  - Filters / Convolutions
  - Pooling layer
  - Architecture of CNN
  - Illustration of what CNNs learn

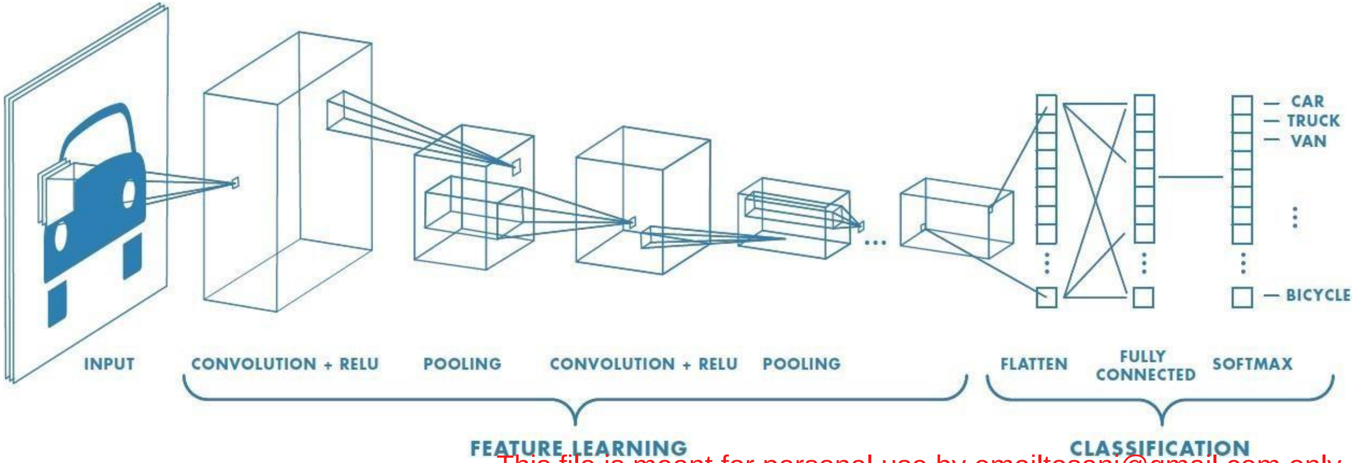
Discussion questions

1. What are convolutional neural networks and how are they different from ANNs?
2. How do filters/kernels work for feature detection in CNNs?
3. How do pooling, padding, and stride operations work in CNNs?
4. What is transfer learning and how can we use that in CNNs?

Convolutional Neural Networks

CNNs are a special type of neural network designed to work with the image data. CNNs use convolutional layers, i.e., hidden layers which perform convolution operations. They have some different characteristics to ANNs:

1. Unlike ANNs, CNNs capture the spatial structure of the image.
2. CNNs follow the concept of parameter sharing, i.e., one filter is applied over the whole image, because of which they are much more computationally efficient.



The first part in this architecture is the convolutional layer followed by the pooling layer and the second part is the fully connected layer. This whole architecture is called a convolutional neural network.

Image Source

The convolutional layer - Filter/Kernel

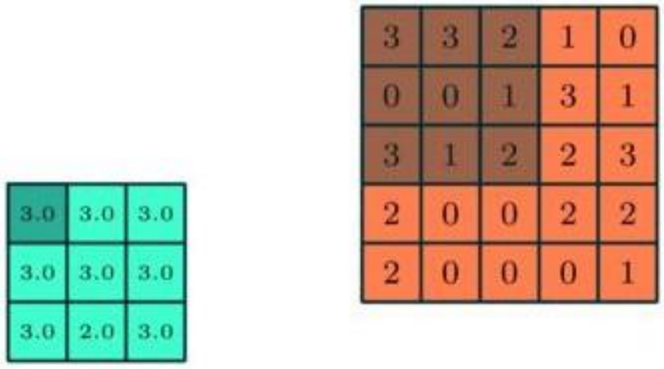
- A convolution operation uses a small array of numbers called a filter/kernel on the input image.
- Each filter is designed to identify a specific feature in the input space of the image, such as horizontal edges, vertical edges, etc.
- A CNN can successfully capture the spatial and temporal dependencies in an image through the application of relevant filters.
- The role of the CNN is to reduce the images into a form which is easier to process, without losing features which are important for getting a good prediction.

- This image shows how the convolution operation works in a CNN
    - It uses a 3x3 filter on a 5x5 image
    - The resulted feature is a 3x3 image which is the convolved feature

Image Source

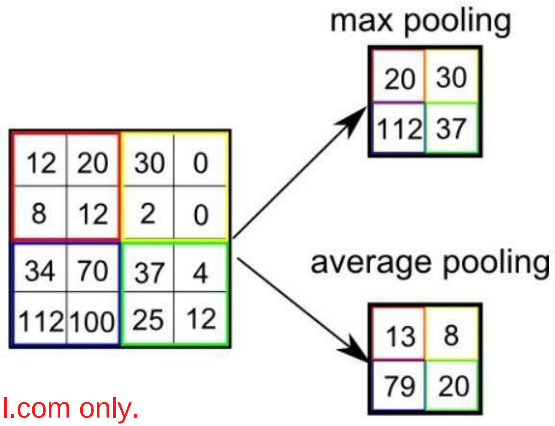
Pooling layer in CNNs

- After a convolution operation, we usually perform pooling to reduce the dimensions of the feature map.
- It enables us to reduce the number of parameters, which both reduces the training time as well as the overfitting.
- Pooling layers downsample each feature map independently, reducing the height and width, but keeping the depth same.



The two most common types of pooling are - Max and Average:

- Max pooling just takes the maximum value, whereas average pooling takes the average value in the pooling window
- Contrary to the convolution operation, pooling has no parameters.
- In these gifs, we can see how max and average pooling work.

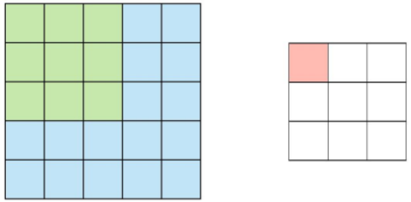


Padding & Stride in CNNs

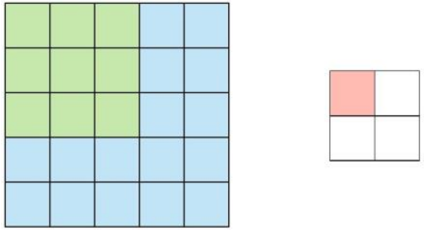
- **Stride** specifies how much we move the filter at each step. By default, the value of the stride is 1 and is represented by the first figure.
- We can also increase the value of stride if we want less overlap between the filters. It also makes the resulting feature map smaller since we are skipping over some locations.
- The second figure demonstrates the stride 2

We see that after using stride 2, the size of the feature map is smaller than the input. If we want to maintain the same dimensions, we can use **padding** to surround the input with zeros.

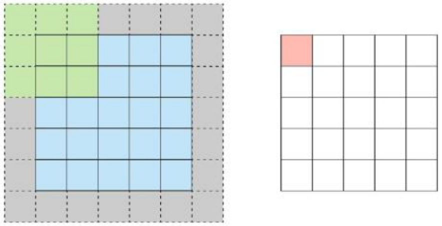
- The grey area around the input in the third figure is the padding.
- We either pad with zeros or the values on the edge, to match the dimensions of the feature map with the input.
- Padding is commonly used in CNNs to preserve the size of the feature maps, otherwise they would shrink at each layer, which is not desirable.



Stride 1 Feature Map



Stride 2 Feature Map



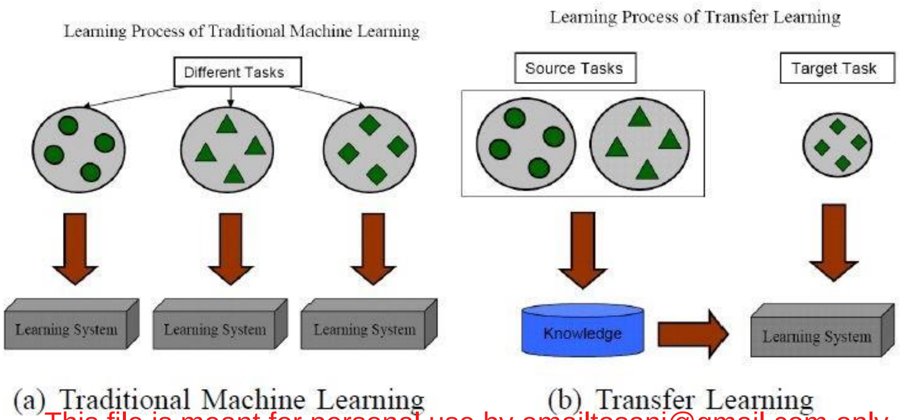
Stride 1 with Padding Feature Map

Topics to be covered

- Transfer Learning

Transfer Learning

- If you are using a specific neural network architecture that has been trained before, you can use these pretrained parameters/weights instead of random initialization to solve the problem.
- It can help boost the performance of the neural network.
- The pretrained models might have trained on large datasets like ImageNet, and taken a lot of time to learn those parameters/weights with optimized hyperparameters. This can save a lot of time.
- If you have enough data, you can fine tune all the layers in your pretrained network but don't randomly initialize the parameters, leave the learned parameters as they are and learn from there.

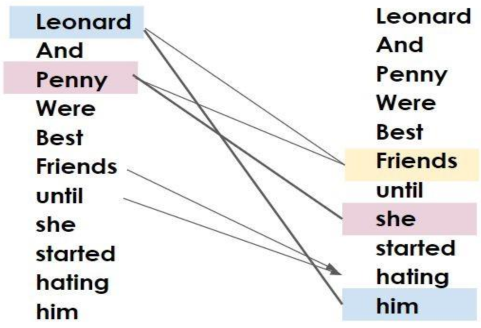


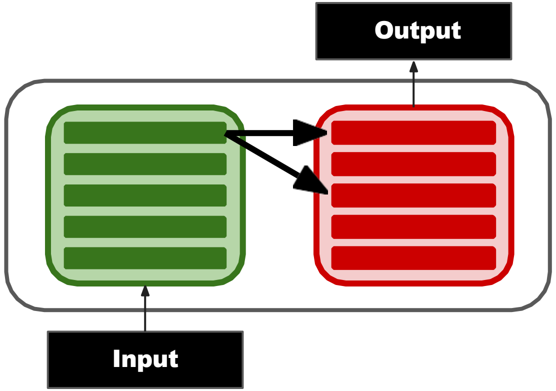
Case Study

Appendix

The Basics of Transformer Models

- Transformers are a **type of neural network architecture**
- Transformers were introduced in a paper by **Vaswani et al. in 2017** and have since become widely used in the NLP community.
- Transformers consist of an encoder and a decoder. The encoder takes in a sequence of tokens (e.g. words or characters) and outputs a sequence **of vectors, called "keys," "values," and "queries."** The decoder then takes these vectors as input and outputs a sequence of tokens.
- Transformers are based on the **idea of self-attention**, which allows the model to consider the entire input sequence when computing the output for each element in the sequence.
- The self-attention mechanism in transformers allows the model to compute the **weighted sum of the values based on the similarity between the queries and keys.** This allows the model to selectively focus on different parts of the input sequence as it processes it.
- BERT GPT, XLNet, T5, Electra are popular transformers





The Need for Transformer Models

The need for transformers arises from the limitations of sequential models, & the desire for models that can capture long-range dependencies and relationships in sequential data, while also being effficient & scalable.

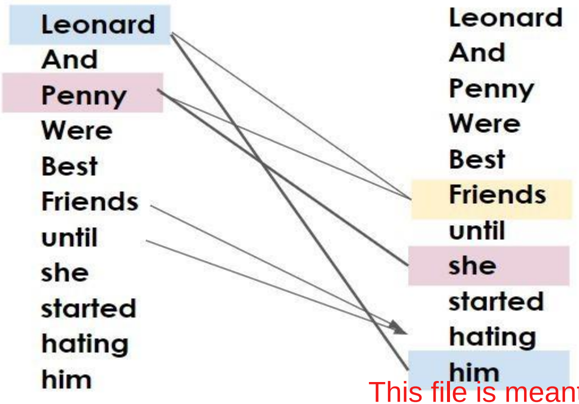
1. They accept the entire corpus of data at once and process it altogether using an attention and positional encoding mechanism, which expresses how one word within the sentence at a particular position is related to all the other words within the same sentence.

2. Due to success of Self attention and Multi-head attention normally massive Transformer models are trained with huge amount of Data. This can perform most diverse tasks mostly in the field of conversational AI while conventional RNNs were limited due to their limitation with parallelizing computations.

The Need for Transformer Models

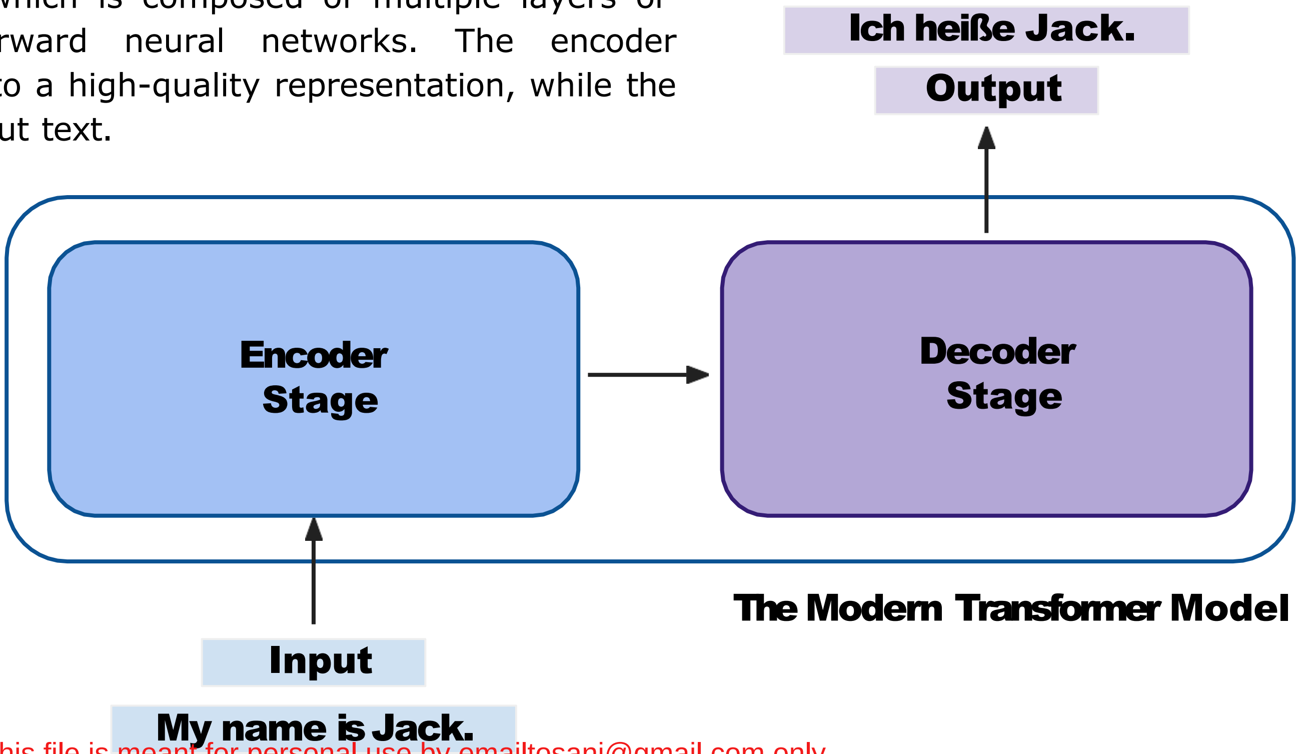
3. They also convert not just words into word embeddings, but sentences into sentence embeddings as well. This is a step-up from word embeddings - these sentence embeddings take care of context and semantic meaning in the sentence.

4. The Transformer uses self-attention and multi-head attention mechanisms to selectively attend to different parts of the input sequence when generating each output token, allowing it to capture complex relationships more effectively.



Basic Transformer Architecture

The basic transformer architecture consists of **an encoder and a decoder stage,** each of which is composed of multiple layers of self-attention and feedforward neural networks. The encoder processes the input text into a high-quality representation, while the decoder generates the output text.



Happy Learning !