CNN Theoretical Questions

Q1: **How are *CNNs* used for Time Series Prediction?**

Answer

- The ability of **CNNs** to learn and automatically extract features from *raw* input data can be applied to time series forecasting problems. A sequence of observations can be treated like a one-dimensional image that a CNN model can read and distill into the most salient elements.
- The capability of *CNNs* has been demonstrated to great effect on time series classification tasks such as automatically detecting human activities from raw accelerator sensor data from fitness devices and smartphones.
- *CNNs* have the support for multivariate input, multivariate output, it can learn arbitrary but complex functional relationships, but does not require that the model learn directly from lag observations. Instead, the model can learn a representation from a large input sequence that is most relevant for the prediction problem.

Source: machinelearningmastery.com

Q2: **What's the difference between *Convolutional Neural Networks (CNN)* and *Recurrent Neural Networks (RNN)* and in which cases would use each one?**

Answer

Convolutional neural nets apply a *convolution* to the data before using it in fully connected layers.

- They are best used in cases where you want *positional invariance*, that is to say, you want features to be captured regardless of where they are in the input sample.
- Think of a picture with all sorts of animals in it. If you apply a *convolutional neural net* to classify whether there is a cat in the picture, it will identify the cat no matter what position in the picture the cat is (at the top, the bottom, left or right). This is very *useful for image classification*.

Recurrent neural nets are neural networks that *keep state* between input samples. They remember previous input samples and use those to help classify the current input sample.

- They are most useful when the *order of your data is important*. So for instance in speech (previous words do help identify the current word), video (frames are ordered) and also text processing.
- Generally speaking, problems related to time-series data (data with a timestamp on them) are good candidates to be solved well with recurrent neural nets.

Source: stats.stackexchange.com

Q3: **How is *Convolutional Neural Networks (CNN)* used in NLP?**

Answer

- **Convolutional neural networks (CNN)** is mostly used in image classification but it can also be used for NLP.
- For NLP tasks the *sentences* are represented as *matrices*. The *row* of the matrix consists of a *token* (or a *character*).
- The filters of the CNN can be made to slide over the *row* of the matrix.
- The *height* may vary, but sliding the windows over `2-5` words is typical.

Source: www.wildml.com

Q4: **Name some advantages of using *Convolutional Neural Networks* vs *Dense Neural Networks* for image classification**

Answer

These are the main advantages of a *CNN* over a *fully connected DNN* for image classification:

- Because consecutive layers are only *partially connected* and because it heavily reuses its weights, a *CNN* has many fewer parameters than a *fully connected DNN*, which makes it much faster to train, reduces the *risk* of overfitting, and requires much less training data.
- When a *CNN* has learned a kernel that can detect a particular feature, it can detect that feature *anywhere* in the image. In contrast, when a *DNN* learns a feature in one location, it can detect it only in that *particular location*. Since images typically have very repetitive features, *CNNs* are able to *generalize* much better than *DNNs* for image processing tasks such as classification, using fewer training examples.
- Finally, a *DNN* has *no prior knowledge* of how pixels are organized; it does not know that nearby *pixels* are close. A *CNN's architecture* embeds this *prior knowledge*. Lower layers typically identify features in small areas of the images, while higher layers combine the lower-level features into larger features. This works well with most natural images, giving *CNNs* a decisive head start compared to *DNNs*.

Source: www.amazon.com

Q5: **In *CNN*, what are the pros and cons of *Max Pooling* vs *Average Pooling*?**

Answer

In a convolutional neural network, the pooling layer helps us to generalize the presence of features. Two popular techniques for this are:

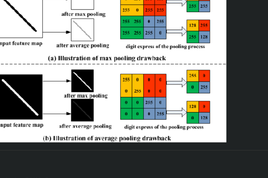
- **Max Pooling**, which retains the *most prominent features* of the feature map.
- **Average Pooling**, which retains the *average values* of features of the feature map.

The above implies that *average pooling* retains a *lot of data*, whereas *max-pooling* rejects a *big chunk* of data.

Hence, average pooling sometimes cannot extract the important features because it takes everything into account, and gives an average value that may or may not be important, meanwhile, max-pooling focuses only on the very important features.

But on other hand, average pooling encourages the network to identify the *complete extent* of the object, whereas max-pooling restricts that to only the *very important* features, and might miss out on some details.

In general, the choice of pooling method is dependent on the data and the *expectations* from the *pooling layer* and the *CNN*.



Source: iq.opengenius.org

Q6: **What do the *fully connected layers* do in *CNNs*?**

Answer

The output from the convolutional layers represents *high-level features* in the data. While that output could be *flattened* and connected to the *output layer*, adding a fully-connected layer is a *(usually) cheap* way of learning non-linear *combinations* of these features.

Essentially the convolutional layers are providing a meaningful, *low-dimensional*, and somewhat invariant feature space, and the fully-connected layer is *learning* a (possibly non-linear) function in that space.

Source: stats.stackexchange.com

Q7: **When would you use *MLP*, *CNN*, and *RNN*?**

Answer

- **Multilayer Perceptrons**, or MLPs for short are the classical type of neural network. They are very flexible and can be used generally to learn a *mapping from inputs to outputs*, however, they are perhaps more suited to *classification* and *regression* problems.
- **Convolutional Neural Networks**, or CNNs, were developed and are best used for *image classification*. But they can also be used generally with data that has a *spatial structure*, such as a sequence of words, and can be used for *document classification*.
- **Recurrent Neural Network** or RNNs, was developed for *sequence prediction* and is well suited for problems that have a sequence of input observations or a sequence of output observations. They are suitable for *text data*, *audio data*, and similar applications.

Source: machinelearningmastery.com

Q8: **How is the *Transformer Network* better than *CNNs* and *RNNs*?**

Answer

- With **RNN**, you have to go *word by word* to access to the cell of the last word. If the network is formed with a long reach, it may take several steps to remember, each masked state (output vector in a word) depends on the previous masked state. This becomes a major problem for GPUs. This sequentiality is an obstacle to the parallelization of the process. In addition, in cases where such sequences are too long, the model tends to forget the contents of the distant positions one after the other or to mix with the contents of the following positions. In general, whenever ***long-term dependencies*** are involved, we know that **RNN** suffers from the ***Vanishing Gradient Problem***.
- Early efforts were trying to solve the dependency problem with ***sequential convolutions*** for a solution to the **RNN**. A long sequence is taken and the convolutions are applied. The disadvantage is that **CNN** approaches require many layers to capture long-term dependencies in the sequential data structure, without ever succeeding or making the network so large that it would eventually become impractical.
- The **Transformer** presents a new approach, it proposes to *encode* each word and apply the ***mechanism of attention*** in order to connect two distant words, then the *decoder* predicts the sentences according to all the words preceding the current word. This workflow can be parallelized, accelerating learning and solving the ***long-term dependencies*** problem.

Source: medium.com

Q9: **Compare the *Convolutional Neural Network* and *Multi-layer Perceptron***

Answer

- **Multi-Layer Perceptron** is a class of feed-forward neural network that consists of at least three layers: an *input layer, a hidden layer, and an output layer.* Except for the input nodes, each node is a neuron that uses a *nonlinear activation function*. MLP utilizes a supervised learning technique called backpropagation for training.
- **CNN** is a type of neural network which is most effective for *computer vision*. It can account for local connectivity where each filter is panned around the entire image according to a certain size and stride, allowing the filter to find and match patterns no matter where the pattern is located in a given image.

- **MLP** is a fully connected network so it has too many parameters. Every node is connected to another node in a very dense web.
- **CNN** is *sparsely* connected rather than fully connected where every node is not connected to every other node.

Source: medium.com

Q10: **What is intuition behind using *CNN* for NLP?**

Answer

- When **CNN** is used for *computer vision*, closer pixels may be semantically related but it is not always true for words. The main reason why a convolution is used in NLP is due to its speed. It is very fast compared to other methods such as *n-grams*.
- **CNNs** are also efficient in terms of representation when large vocabularies are present.
- **CNNs** used for classification tasks seem to be the most natural fit, such as
  - *Sentiment Analysis*,
  - *Spam Detection*, or
  - *Topic Categorization*.

Source: www.wildml.com

Q11: **What's the difference between *multi-headed* and *multi-channel* CNNs?**

Answer

- A **multi-headed** CNN is a model that has more than *one input* or "*head*" for reading input. It often allows the model to access the same input image multiple times using a different *sized kernel*, and in turn, allows different features to be extracted in parallel from the data.

- A **multi-channel** CNN is a model that receives input that has *multiple variables* or "*channels*", such as red, green and blue channels for an image for a `2D` CNN, or parallel time series in the case of a `1D` CNN. All channels will be read together using the same filters.

Source: machinelearningmastery.com

Q12: **What's the difference between *CNN-LSTMs* and *ConvLSTMs*?**

Answer

Consider a sequence prediction problem where we wish to extract features from each timestep before processing the sequence of extracted features. Two examples include:

- `1D` : Sequence of daily subsequences of observations over a month in a *time series* prediction problem.
- `2D` : Sequence of still images in the case of *video classification*.

Data of this form can be addressed with a **CNN-LSTM** or a **ConvLSTM model**.

- The **CNN-LSTM** will use a *CNN model* to extract features from each step in the input sequence, resulting in a *sequence of extracted features*. This sequence of extracted features can be interpreted by an *LSTM model*.

- The **ConvLSTM** is different in that *each step* in the input sequence is processed by *LSTM units directly* using a convolutional operation inside the unit, not as a *feature extraction* step beforehand and passed as input to the unit as in the *CNN-LSTM*.

Source: machinelearningmastery.com

Q13: **What's the difference between *Convolutional Layers* vs *Fully Connected Layers*?**

Answer

- A **convolutional layer** applies the *same filter* repeatedly at *different positions* in the layer below it. E.g. if the input layer has dimensions `512 x 512` , you could have a convolutional layer that applies the same `8 x 8` filter (specified by `64` filter coefficients), at *each point* in (e.g.) a `128 x 128` grid overlaid on the input layer.

- On the other hand, in a **fully connected layer**, *each node* would learn `512 x 512` weights, one for each of the nodes in the input layer.

- Therefore **convolutional layers** are well suited to detect **local features** that may **appear anywhere** in the input (e.g. edges in a visual image). The idea is that you don't have to train every node separately to detect the same feature, but rather you learn one filter that is shared among all the nodes.

- **Fully connected layers** are used to detect **specific global configurations** of the features detected by the lower layers in the net. They usually sit at the top of the *network hierarchy*, at a point when the input has been reduced (by the previous, usually convolutional layers) to a compact representation of features. Each node in the FC layer learns its own set of weights on all of the nodes in the layer below it.

- So you can (roughly) think of **convolutional layers** as *breaking the input* (e.g. an image) up into common features, and the **FC layers** as *piecing those features together* into e.g. objects that you want the network to recognize.

Source: stats.stackexchange.com

CNN Practical Challenges

Q1: **Describe the architecture of a typical Convolutional Neural Network (CNN)**

Answer

- In a typical *CNN* architecture, a few *convolutional layers* are connected in a cascade *style*.
- Each *convolutional layer* is followed by a *Rectified Linear Unit* (ReLU) layer, then a *pooling layer*, then one or more convolutional layers (+*ReLU*), then another *pooling layer*.
- The output from each *convolution layer* is a set of objects called *feature maps*, generated by a single *kernel filter*.
- The *feature maps* are used to define a new input to the next layer.
- At the end, there is one or more *fully connected layers*.
- Pretty much depending on problem type, the network might be *deep* though.

Source: www.amazon.com