**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **17-06-2020** | **Name:** | **Karthik J** |
| **Course:** | Cyber Security Go from Zero | **USN:** | **4AL16EC030** |
| **Topic:** |  | **Semester & Section:** | **8TH A** |
| **GitHub Repository:** | Karthik-J |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **OSI Models**  The open system interconnection model, better known as the [OSI model](https://www.techopedia.com/definition/24205/open-systems-interconnection-model-osi-model), is a [network map](https://www.techopedia.com/definition/4993/network-map) that was originally developed as a universal standard for creating networks. But instead of serving as a model with agreed-upon protocols that would be used worldwide, the OSI model has become a teaching tool that shows how different tasks within a network should be handled in order to promote error-free [data transmission](https://www.techopedia.com/definition/9756/data-transmission).  These jobs are split into seven layers, each of which depends on the functions “handed-off” from other layers. As a result, the OSI model also provides a guide for troubleshooting network problems by tracking them down to a specific layer. Here we’ll take a look at the layers of the OSI model and what functions they perform within a network. 1. Physical Layer The [physical layer](https://www.techopedia.com/definition/8866/physical-layer) is the actual cable, fibers, cards, switches and other mechanical and electrical equipment that make up a network. This is the layer that transforms digital data into signals that can be sent down a wire to transmit data. These signals are often electrical but, as in the case of [fiber optics](https://www.techopedia.com/definition/14931/fiber-optic), they can also be non-electrical signals such as optics or any other type of pulse that can be digitally encoded. From a networking perspective, the purpose of the physical layer is to provide the architecture for data to be sent and received. The physical layer is probably the easiest layer to troubleshoot but the most difficult to repair or construct, as this involves getting the hardware infrastructure hooked up and plugged in. 2. Data Link Layer The [data link layer](https://www.techopedia.com/definition/18698/data-link-layer) is where information is converted into coherent [“packets”](https://www.techopedia.com/definition/5380/packet) and frames that are passed to higher layers. Essentially, the data link layer unpacks [raw data](https://www.techopedia.com/definition/1230/raw-data) coming in from the physical layer and translates information from the upper layers into raw data to be sent over the physical layer. The data link layer is also responsible for catching and compensating for any errors that occur in the physical layer. 3. Network Layer The [network layer](https://www.techopedia.com/definition/24204/network-layer) is where the destination for incoming and outgoing data is set. If the data link layer is the highway for cars to drive on, the network layer is the GPS system telling drivers how to get there. Addressing is added to the data by tacking on information around the data packet in the form of an address header. This layer is also responsible for determining the quickest route to the destination and the handling of any problems with [packet switching](https://www.techopedia.com/definition/5603/packet-switching) or network congestion. This is the layer where routers work to ensure that data is properly re-addressed before passing it on to the next leg of the packet’s journey. 4. Transport Layer The [transport layer](https://www.techopedia.com/definition/9760/transport-layer) is responsible for streaming data across the network. At this level, the data is not thought of in terms of individual packets but more in terms of a conversation. To accomplish this, protocols – which are defined as “rules of communication” – are used. The protocols watch the complete transmission of many packets – checking the conversation for errors, acknowledging successful transmissions and requesting retransmission if errors are detected.  The network layer and the transport layer work together like a postal system. The network layer addresses the data, much like a person addresses an envelope. Then, the transport layer acts as the sender’s local postal branch, sorting and grouping all similarly addressed data into larger shipments bound for other local branches, where they will then be delivered. 5. Session Layer The [session layer](https://www.techopedia.com/definition/9322/session-layer) is where connections are made, maintained and ended. This usually refers to application requests for data over the network.  Whereas the transport layer handles the actual flow of data, the session layer acts as an announcer, making sure that the programs and applications requesting and sending data know their requests are being filled. In technical terms, the session layer synchronizes data transmission. 6. Presentation Layer The [presentation layer](https://www.techopedia.com/definition/8955/presentation-layer) is where received data is converted into a format that the application it is destined for can understand. The work done at this layer is best understood as a translation job. For example, data is often [encrypted](https://www.techopedia.com/definition/5507/encryption) at the presentation layer before being passed to the other layers for sending. When data is received, it will be [decrypted](https://www.techopedia.com/definition/1773/decryption) and passed on to the application it is intended for in the format that is expected. 7. Application Layer The [application layer](https://www.techopedia.com/definition/6006/application-layer) coordinates network access for the software running on a particular computer or device. The protocols at the application layer handle the requests that different software applications are making to the network. If a web browser wants to download an image, an email client wants to check the server and a file-sharing program wants to upload a movie, the protocols in the application layer will organize and execute these requests. TCP/IP model  * The TCP/IP model was developed prior to the OSI model. * The TCP/IP model is not exactly similar to the OSI model. * The TCP/IP model consists of five layers: the application layer, transport layer, network layer, data link layer and physical layer. * The first four layers provide physical standards, network interface, internetworking, and transport functions that correspond to the first four layers of the OSI model and these four layers are represented in TCP/IP model by a single layer called the application layer. * TCP/IP is a hierarchical protocol made up of interactive modules, and each of them provides specific functionality. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date:** | | **17-06-2020** | **Name:** | **Karthik J** |  |
| **Course:** | | CNN for Computer Vision with Keras and TensorFlow in Python | **USN:** | **4AL16EC030** |  |
| **Topic:** | |  | **Semester & Section:** | **8th A** |  |
|  | **AFTERNOON SESSION DETAILS** | | | | |
|  | **My work:** <https://github.com/Karthikjsannakki/CNN>  **Image of session** | | | | |
|  | **Convolutional neural network** In [deep learning](https://en.wikipedia.org/wiki/Deep_learning), a **convolutional neural network** (**CNN**, or **ConvNet**) is a class of [deep neural networks](https://en.wikipedia.org/wiki/Deep_neural_network), most commonly applied to analyzing visual imagery. They are also known as **shift invariant** or **space invariant artificial neural networks** (**SIANN**), based on their shared-weights architecture and [translation invariance](https://en.wikipedia.org/wiki/Translation_invariance) characteristics. They have applications in [image and video recognition](https://en.wikipedia.org/wiki/Computer_vision), [recommender systems](https://en.wikipedia.org/wiki/Recommender_system), [image classification](https://en.wikipedia.org/wiki/Image_classification), [medical image analysis](https://en.wikipedia.org/wiki/Medical_image_computing), [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), and financial [time series](https://en.wikipedia.org/wiki/Time_series).  CNNs are [regularized](https://en.wikipedia.org/wiki/Regularization_(mathematics)) versions of [multilayer perceptrons](https://en.wikipedia.org/wiki/Multilayer_perceptron). Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to [overfitting](https://en.wikipedia.org/wiki/Overfitting) data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.  Convolutional networks were [inspired](https://en.wikipedia.org/wiki/Mathematical_biology) by [biological](https://en.wikipedia.org/wiki/Biological) processes in that the connectivity pattern between [neurons](https://en.wikipedia.org/wiki/Artificial_neuron) resembles the organization of the animal [visual cortex](https://en.wikipedia.org/wiki/Visual_cortex). Individual [cortical neurons](https://en.wikipedia.org/wiki/Cortical_neuron) respond to stimuli only in a restricted region of the [visual field](https://en.wikipedia.org/wiki/Visual_field) known as the [receptive field](https://en.wikipedia.org/wiki/Receptive_field). The receptive fields of different neurons partially overlap such that they cover the entire visual field.  CNNs use relatively little pre-processing compared to other [image classification algorithms](https://en.wikipedia.org/wiki/Image_classification). This means that the network learns the [filters](https://en.wikipedia.org/wiki/Filter_(signal_processing)) that in traditional algorithms were [hand-engineered](https://en.wikipedia.org/wiki/Feature_engineering). This independence from prior knowledge and human effort in feature design is a major advantage. Architecture A convolutional neural network consists of an input and an output layer, as well as multiple [hidden layers](https://en.wikipedia.org/wiki/Multilayer_perceptron#Layers). The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other [dot product](https://en.wikipedia.org/wiki/Dot_product). The activation function is commonly a [RELU layer](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)), and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final [convolution](https://en.wikipedia.org/wiki/Convolution).  Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a sliding dot product or [cross-correlation](https://en.wikipedia.org/wiki/Cross-correlation). This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point. Convolutional When programming a CNN, the input is a [tensor](https://en.wikipedia.org/wiki/Tensor) with shape (number of images) x (image height) x (image width) x ([image depth](https://en.wikipedia.org/wiki/Image_depth)). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map height) x (feature map width) x (feature map channels). A convolutional layer within a neural network should have the following attributes:   * Convolutional kernels defined by a width and height (hyper-parameters). * The number of input channels and output channels (hyper-parameter). * The depth of the Convolution filter (the input channels) must be equal to the number channels (depth) of the input feature map.   Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its [receptive field](https://en.wikipedia.org/wiki/Receptive_field). Although [fully connected feedforward neural networks](https://en.wikipedia.org/wiki/Multilayer_perceptron) can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. By using regularized weights over fewer parameters, the vanishing gradient and exploding gradient problems seen during [backpropagation](https://en.wikipedia.org/wiki/Backpropagation) in traditional neural networks are avoided.[[14]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-14)[[15]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-15) Pooling Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. In addition, pooling may compute a max or an average. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Average pooling uses the average value from each of a cluster of neurons at the prior layer. Fully connected Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional [multi-layer perceptron](https://en.wikipedia.org/wiki/Multi-layer_perceptron) neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images. Receptive field In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its receptive field. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer. The subarea of the original input image in the receptive field is increasingly growing as getting deeper in the network architecture. This is due to applying over and over again a convolution which takes into account the value of a specific pixel, but also some surrounding pixels. Weights Each neuron in a neural network computes an output value by applying a specific function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning, in a neural network, progresses by making iterative adjustments to these biases and weights.  The vector of weights and the bias are called filters and represent particular [features](https://en.wikipedia.org/wiki/Feature_(machine_learning)) of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces [memory footprint](https://en.wikipedia.org/wiki/Memory_footprint) because a single bias and a single vector of weights are used across all receptive fields sharing that filter, as opposed to each receptive field having its own bias and vector weighting. **Keras** [tf.keras](https://www.tensorflow.org/api_docs/python/tf/keras) is TensorFlow's high-level API for building and training deep learning models. It's used for fast prototyping, state-of-the-art research, and production, with three key advantages:   * User-friendly Keras has a simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors. * Modular and composable Keras models are made by connecting configurable building blocks together, with few restrictions. * Easy to extend Write custom building blocks to express new ideas for research. Create new layers, metrics, loss functions, and develop state-of-the-art models.  **Seaborn: Python's Statistical Data Visualization Library** One of the best but also more challenging ways to get your insights across is to visualize them: that way, you can more easily identify patterns, grasp difficult concepts or draw the attention to key elements. When you’re using Python for data science, you’ll most probably will have already used [Matplotlib](https://matplotlib.org/), a 2D plotting library that allows you to create publication-quality figures. Another complimentary package that is based on this data visualization library is [Seaborn](http://seaborn.pydata.org/), which provides a high-level interface to draw statistical graphics. | | | | |