 ————— o —————



**Module 23**

**Neural Networks for Vision**

* [Video Transcripts](https://student.emeritus.org/courses/4765/files/3652701?wrap=1)
* [Download Video Transcripts](https://student.emeritus.org/courses/4765/files/3652701/download?download_frd=1)
* [Quick Reference Guide](https://student.emeritus.org/courses/4765/files/3652702?wrap=1)

**Glossary**

**Convolutional Neural Network (CNN)**

An algorithm that processes deep learning input images by assigning importance to weights and biases for various aspects and objects in the images; its structure reflects the connectivity pattern of neurons in the human brain and is based on the organization of the visual cortex

**Data Augmentation**

A technique that involves adding distorted, rescaled, and rotated versions of each image to the training set

**Fully Connected (FC) Layer**

A building block of a CNN that usually forms the last few layers of a CNN; the input for the FC layer is the output from the final pooling layer

**Long Short-Term Memory Network (LSTM)**

A type of RNN that is capable of learning long-term dependencies in sequence prediction problems

**Pooling Layer**

A building block of a CNN that compresses the information down to a smaller size to reduce the number of parameters

**Recurrent Neural Network (RNN)**

A type of neural network that works well for machine learning problems involving sequential data

**Spatial Correlation**

The concept that a pixel will likely be similar to its neighboring pixels

**Notes:**

**Tricks of the Trade**

Researchers and practitioners acquire knowledge through experience and word-of-mouth, which helps them successfully apply neural networks to complex real-world problems. Unfortunately, the ‘tricks’ they learn, although often theoretically sound, can take years for newcomers to assimilate. As a result, newcomers to the field spend a lot of time wondering why their networks train slowly and perform poorly. This mini-lesson will help you acquire some techniques to assist you in preparing high-performance neural networks. While there are many strategies, this mini-lesson will focus on those relevant to this module.

**Shuffling the Examples**

Neural networks learn the fastest from the most random samples. Therefore, selecting a sample that is the most unfamiliar to the system at each iteration is advisable. However, this applies only to stochastic learning (since the batch does not care about the order of the presentation of inputs). Although there is no simple way to determine which inputs are information rich, one straightforward technique that crudely implements this idea is to choose consecutive examples from different classes (since training examples belonging to the same class will most likely contain similar information).

**Early Stopping**

Regularization techniques are essential to improve the generalization ability of neural networks. One of the most commonly used strategies is early stopping. This technique can be explained in its simplest form as follows:

Take a validation set that is independent of the training set and monitor the errors on this set during training. In the training set, the error will decrease, whereas, in the validation set, the error will decrease and then increase. The early stopping point occurs when the validation set errors are at their lowest. The network weights are at their most general at this point.

**Neural Networks Classification and Prior Class Probabilities**

A common problem in multilayer perceptron (MLP) classification is related to the prior probabilities of the individual classes. If the number of training examples that correspond to each class differs significantly, then sometimes, the network may have difficulty learning the rarer classes. When prior class probabilities are unequal, a simple method of alleviating difficulty is to adjust (e.g., equalize) the number of patterns in each class. You can do this either through subsampling (removing patterns from higher frequency classes), or by duplication (of patterns in lower frequency classes). In subsampling, patterns may be removed randomly, or heuristics may be used to remove patterns in regions with low ambiguity. However, subsampling can cause lost information.

**Applying Divide and Conquer to Large-Scale Pattern Recognition Tasks**

Traditionally, large-scale problems have always been solved using the divide-and-conquer paradigm, which is a powerful approach. Hence, a hierarchical approach can be used to modularize classification tasks in large-scale application domains, such as speech recognition, where there are thousands of classes and millions of training samples. Divide and conquer proves to be an effective tool for breaking down a complex problem into many smaller tasks. Furthermore, agglomerative clustering can automatically impose a suitable hierarchical structure on a set of classes, even if it contains tens of thousands of classes. In contrast to the relatively small standard benchmarks for learning machines, factors such as training method, model selection, and generalization ability take on a new meaning when dealing with large-scale probability estimation problems.

**Neural Networks for Text or Time Series**

The use of neural networks for time series forecasting has been widely adopted. Most often, these are feed-forward networks that use a sliding window over the input sequence. Examples include market predictions, meteorological forecasting, and network traffic forecasting. Two of the most popular models used for time series are convolutional neural networks (CNNs) and long short-term memory networks (LSTMs).

**Convolutional Neural Networks (CNNs)**

CNNs can be used to predict time series from raw input data by learning and automatically extracting features. For example, an observation sequence can be treated as a one-dimensional image from which a CNN model can interpret and extract the salient elements.

**Long Short-Term Memory Networks (LSTMs)**

LSTMs, which add explicit order handling to the learning of a mapping function from inputs to outputs, are unavailable with MLPs or CNNs. They are neural networks that support sequences of observations as input data.

**Module Issues:**

23.1 problems failing with errors

23.2 **Problem 3** Failing

23.4 problems failing with errors

23.5 problems failing with errors

**Quizes:**

If all the neighbors of a particular pixel are white, then it is likely that the pixel itself is also white. This concept is called spatial correlation. : True

*You are correct! The answer “*True*” is correct because if all the neighboring pixels are white, it is likely that the pixel itself is also white, and this concept is called spatial correlation.*

Which of the following is not a component in the filter of a CNN? : Constant

*You are correct! The answer “*Constant*” is correct because this is not a component in the filter of a CNN.*

The feature map indicates the locations in an image where the feature that is not represented by the filter can be found. : False

*You are correct! The answer “*False*” is correct because the feature map* *indicates the locations in an image where the feature that is represented by the filter can be found.*

In Keras, there is a parameter for padding, which can be set to (blank) if no padding is desired. : Valid

*You are correct! The answer “*Valid*” is correct because, in Keras, the padding parameter can be set to ‘valid’ if no padding is desired.*

The max pooling operation takes a feature map as input and produces another feature map of (blank). : Reduced size

*You are correct! The answer “*Reduced size*” is correct because the pooling layer compresses the information down to a smaller size.*

Which of the following is not a part of specifying a convolution layer in Keras? : Flatten and softmax

*You are correct! The answer “*Flatten and softmax*” is correct because this is* *not a part of specifying a convolution layer in Keras.*

What is the statement in the Keras library for building a convolution layer? : layers.Conv2D(filters,kernel\_size,activation)

*You are correct! The answer “*layers.Conv2D(filters,kernel\_size,activation)*” is correct because this is the statement for building a convolution layer.*

The max pooling layer used after a convolution layer in the Keras library is

layers.Pooling2D(pool\_size) : False

*You are correct! The answer “*False*” is correct because the correct Python statement for the max pooling layer is*“layers.MaxPooling2D(pool\_size)”.

The first convolution layer takes as input a 28 by 28 by 1 image, 32 filters of a 3 by 3 grid, and produces an output of size 26 by 26 by 32. To do this, it requires (blank) parameters. : 320 [(( 3 x 3 ) + 1 ) \* 32]

*You are correct! The answer “*320*” is correct because each of the 32 filters is a 3 by 3 grid of numbers, meaning you have (3 × 3 + 1) × 32 parameters. The extra plus one is from the bias term for each filter.*

The pooling layer in CNNs requires zero parameters. : True

*You are correct! The answer “*True*” is correct because there are no parameters in a max pooling layer. This layer type does a specific predefined task, which requires no training.*

What function in the keras.utils.vis\_utils library plots neural networks? : plot\_model()

*You are correct! The answer “*plot\_model()*” is correct because this is the function used to plot neural networks.*

What is the technique where you add a bunch of distorted, rescaled, and rotated versions of each image in the training set? : Data augmentation

*You are correct! The answer “*Data augmentation*” is correct because this is the technique where you add a bunch of distorted, rescaled, and rotated versions of each image in the training set.*

The technique in which you randomly disable connections between some neurons is called “dropout.” : True

*You are correct! The answer “*True*” is correct because the technique where you randomly disable connections between some neurons is called “dropout”.*

The input to a neural network is considered the top, and the output of a neural network is considered the bottom. : False

*You are correct! The answer “*False*” is correct because the input to a neural network is considered the bottom, and the output of a neural network is considered the top.*

What is the statement that is used in Python to freeze the vgg16 layer for the variable conv\_base? : conv\_base.trainable = False

*You are correct! The answer “*conv\_base.trainable = False*” is correct because this is the statement used to freeze the vgg16 layer for the variable conv\_base.*

What do you call the function that can manipulate images so that they also have the quirks of the dataset that was originally fed to vgg16? : keras.applications.vgg16.preprocess\_input()

The answer *“*keras.applications.vgg16.preprocess\_input()*” is correct because this is the function that can manipulate images so that they also have the quirks of the dataset that was originally fed to vgg16.*

The correct Python statement used to load the vgg16 model with a chopped output layer is:

conv\_base = keras.applications.vgg16.VGG16(

    weights="imagenet"'

    include\_top=True,

    input\_shape=(224, 224, 3))

: False

*You are correct! The answer “*False*” is correct because the parameter that chops the output layer from the vgg16 model is “Include\_top = false”.*

The following code represents the first of the three steps involved in fine-tuning a model:

history\_with\_vgg16\_bottom = model\_with\_vgg16\_bottom.fit(

    train\_dataset, epochs=30,

    validation\_data=validation\_dataset)

: True

*You are correct! The answer "*True*" is correct because the code shown represents the first step in the process of fine-tuning a model.*

When fine-tuning a model, you should unfreeze (blank). : The top-most layers

*You are correct! The answer "*The top-most layers*" is correct because these layers capture high-level features, which capture higher-level features that are more likely to be usefully informed by the dataset.*

What is the type of neural network that works best for sequential data? : Recurrent neural network

*You are correct! The answer “*Recurrent neural network*” is correct because this is the type of neural network that works best for sequential data.*

The total RNN memory content at any point in time is called the memory cell of the network. : False

*You are correct! The answer “*False*” is correct because the total RNN memory content at any point in time is called the state of the network.*

What is the Python statement that adds an LSTM input layer with 16 internal units? : model.add(LSTM(16, input\_shape=(1,6)))

*You are correct! The answer “*model.add(LSTM(16, input\_shape=(1,6)))*” is correct because this is the Python statement that adds an LSTM input layer with 16 internal units.*

In the memory cell, a unit — known as the memory storage unit — stores a copy of the previous output. : True

*You are correct! The answer “*True*” is correct because the memory storage unit stores a copy of the previous output.*

Which activation function allows you to use neural networks for regression problems? : None

*You are correct! The answer “*None*” is correct because this is the activation function that allows you to use neural networks for regression problems.*

Which two statements can be used in Python to build a neural network for a regression problem? : Model = keras.sequential([layers.dense(64,activation=”relu”), layers.dense(1,activation=None)])

and

Model = keras.sequential([layers.dense(64,activation=”relu”), layers.dense(1)]) Model = keras.sequential([layers.dense(64,activation=”relu”), layers.dense(1,activation=None)])

*You are correct! The answers “*Model = keras.sequential([layers.dense(64,activation=”relu”),

layers.dense(1)])*”, “*Model = keras.sequential([layers.dense(64,activation=”relu”),

layers.dense(1,activation=None)])*” are correct because the final layer has an activation of “None”. Note that you can also simply not provide an activation and the default of None will be used.*

**Try-It Activity 23.1: Visualizing Convolutional Layers - Section B**

For this exercise, I dowloaded an image of red fox and used ResNet50 CNN model to predict:



It correctly predicted red fox as follows:

red\_fox (87.62%)

kit\_fox (9.92%)

dhole (1.53%)

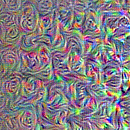
However, when I tried it with ResNet50V2, it falsely predicted as "english foxhound" with 100% confidence as opposed to my expectation of better prediction:(

**Visualizing convnets**

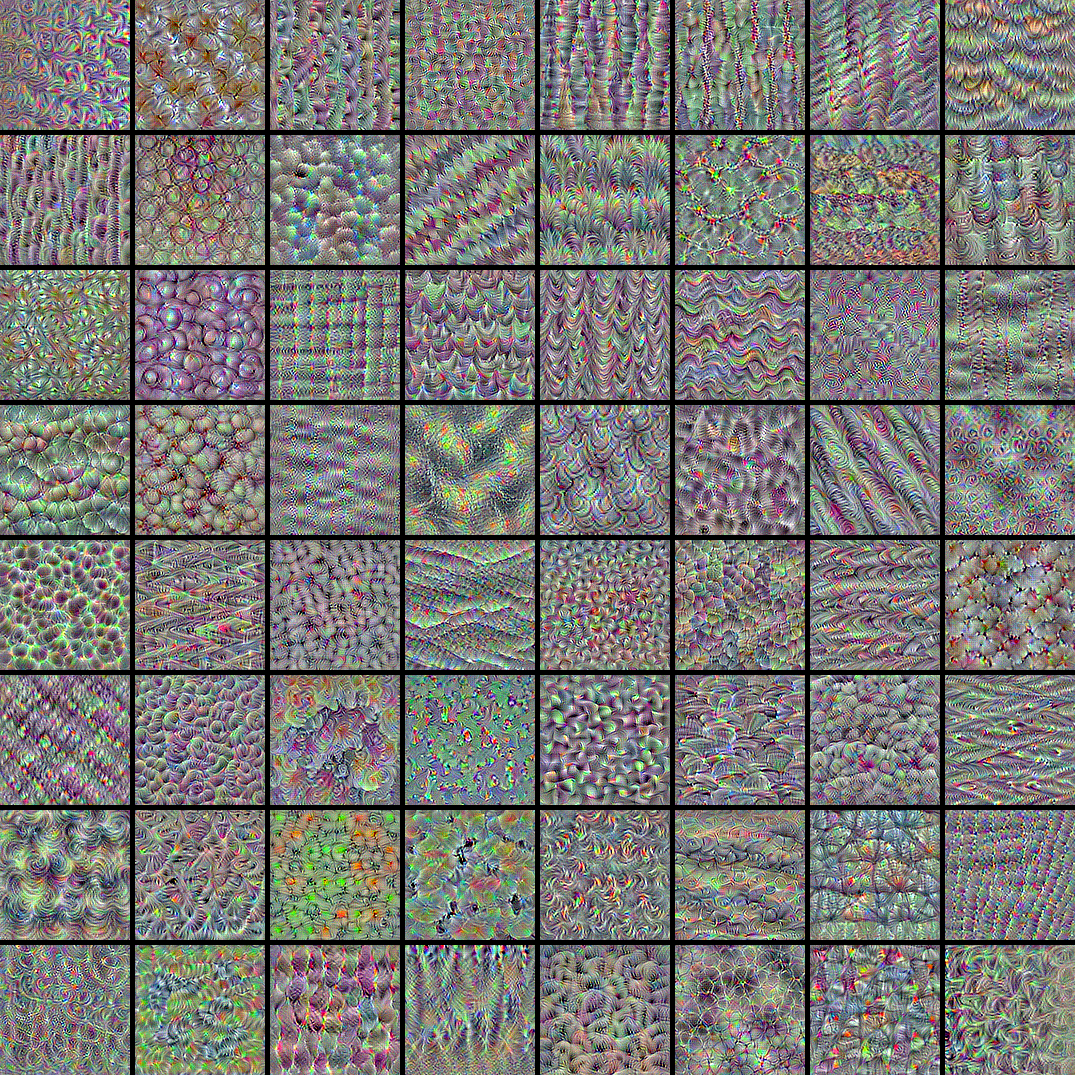
Used  pre-trained model - **ResNet50V2**

**Target Layer -** **conv4\_block2\_out**

**Filter 0**



**8x8 Grid of 1st 64 Filters**



**Try-It Activity 23.2: Sequential Data Models with Keras - Section B**

I used a dataset contains 9357 records of air pollutants with hourly averaged responses from 5 sensors, the measurement was taken for one year at road level in an Italian town: <https://archive.ics.uci.edu/ml/datasets/Air+Quality>. I used this set earlier in try-it 10.2 which did not perform well. I only used CO reading, I did some cleanup as follows:

1. Made CO column float after replacing decimal symbol to ‘.’ from ‘,’: airq['CO'] = airq['CO(GT)'].str.replace(',','.').astype('float')
2. Missing measurement marked as -200, replaced them with mean() value: airq[airq['CO'] == -200] = 2.15275
3. Transformed date and time columns to airq['datetime'] = pd.to\_datetime(aq['Date'] + ' ' + aq['Time'], format='%d/%m/%Y %H.%M.%S') and set index

Build historical and future datasets Split last 4 days (96 hours) for future prediction and rest in history.

I built an LSTM - Longterm short term memory model:

# Model

look\_back = 24

model = Sequential()

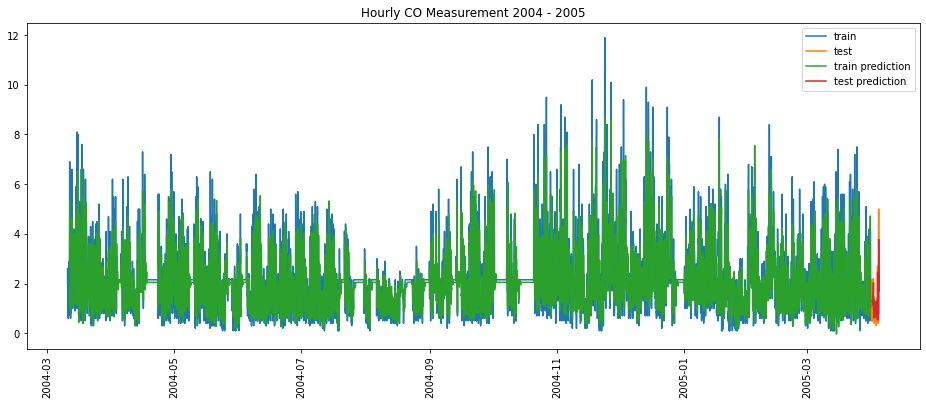
model.add(LSTM(20, input\_shape=(1, look\_back)))

model.add(Dense(1))

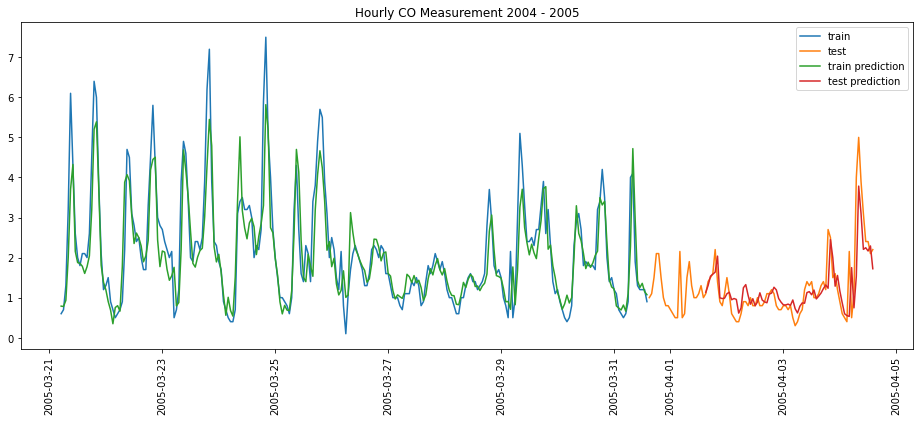
model.compile(loss='mean\_squared\_error', optimizer='adam')

model.fit(X\_train, Y\_train, epochs=20, batch\_size=1, verbose=2)

Which turned out to be a better predicting model than 10.2 try-it certainly:



A close up to last 150 points:



Train Score: 0.65 RMSE

Test Score: 0.53 RMSE

The gap due to 24 data points look back which can be prevented by overlapping the 24 data points between train and test datasets.

Test MSE is pretty low, suggesting the model is performing well. I have not tried GRU, not sure how much more it could improve the results.

**Discussion 23.1: Personal Application - Section B**

I work in software engineering specifically on customer interactions by maintaining a data store just for events and customer journey trail maps.

**1. Classification**

Analyzing customer behavior to understand the customer sentiments.

**2. Regression**

Predicting next possible customer interaction using customer interactions history.

**3. Neural Network**

Neural Network can be used to increase accuracy in time series, regression and classification to detect customer anomalies as well as helping out customers on track for their next steps in interactions.