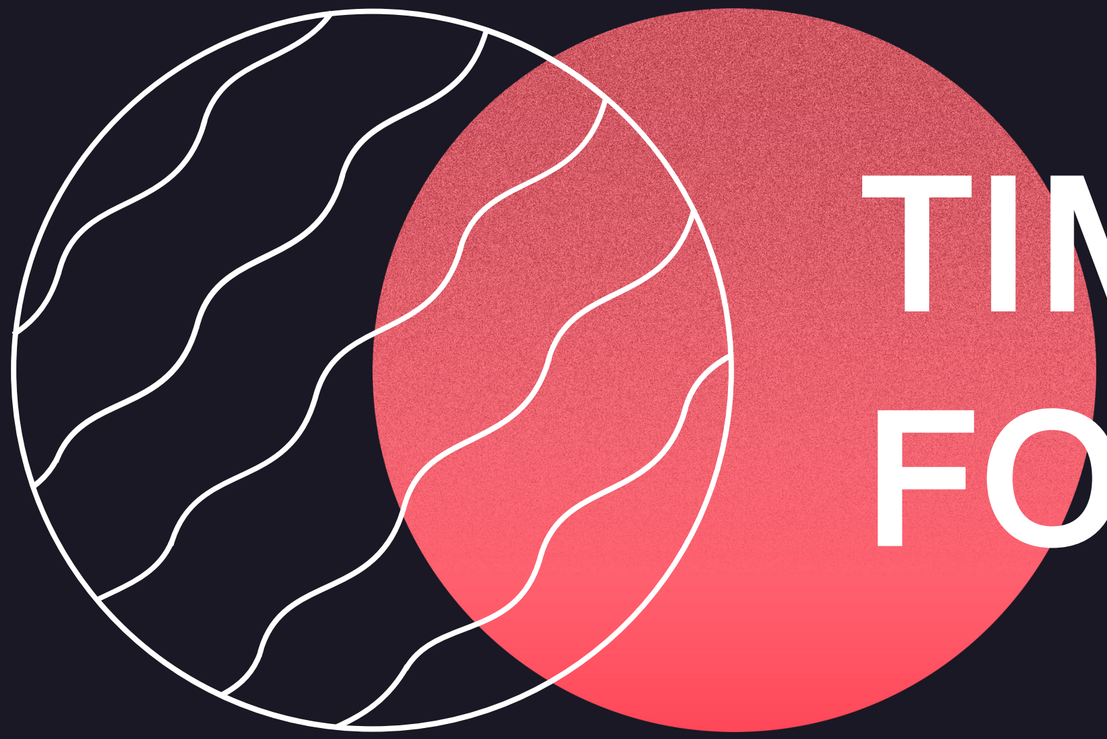


TIME SERIES FORCASTING

SHAPEAI

COMMON PATTERNS IN TIME SERIES

TREND

The trend shows the general tendency of the data to increase or decrease during a long period of time.

SEASONALITY

Seasonality is the presence of variations that occur at specific regular intervals. Seasonal fluctuations in a time series can be contrasted with cyclical patterns.

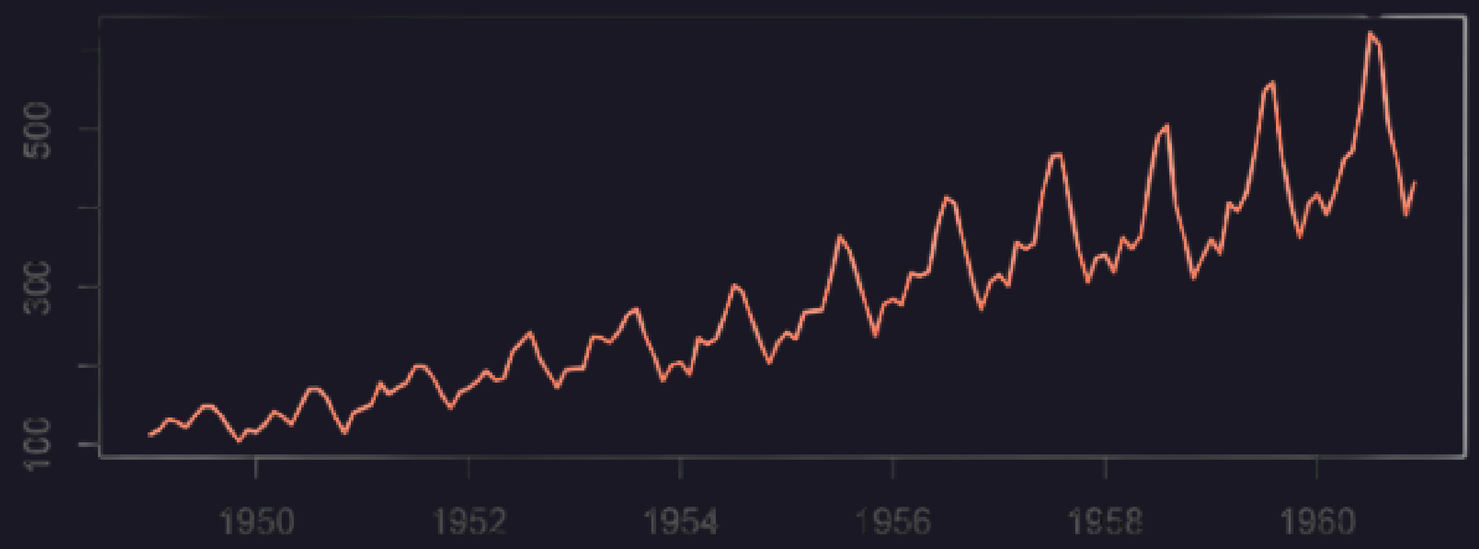
NOISE

A time series is white noise if the variables are independent and identically distributed with a mean of zero. This means that all variables have the same variance zero.

03

TREND

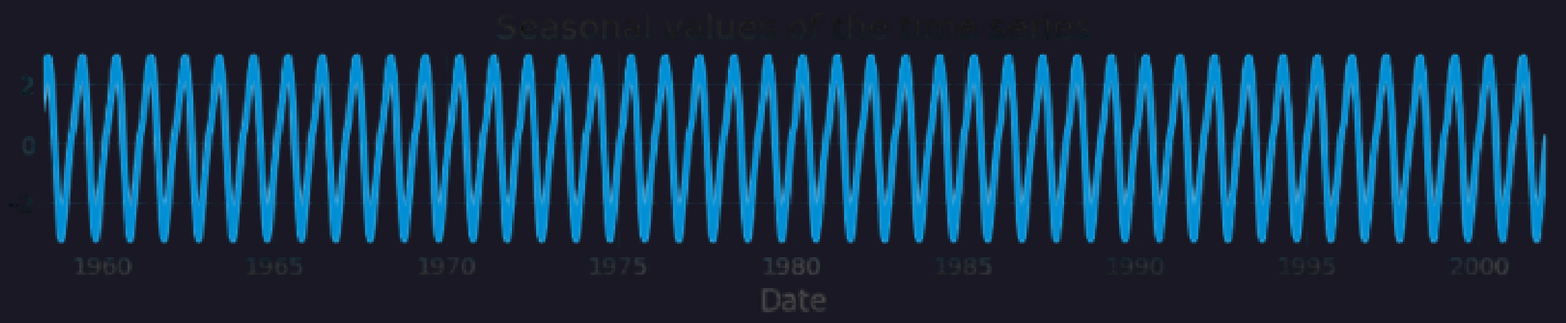
The trend shows the general tendency of the data to increase or decrease during a long period of time.



04

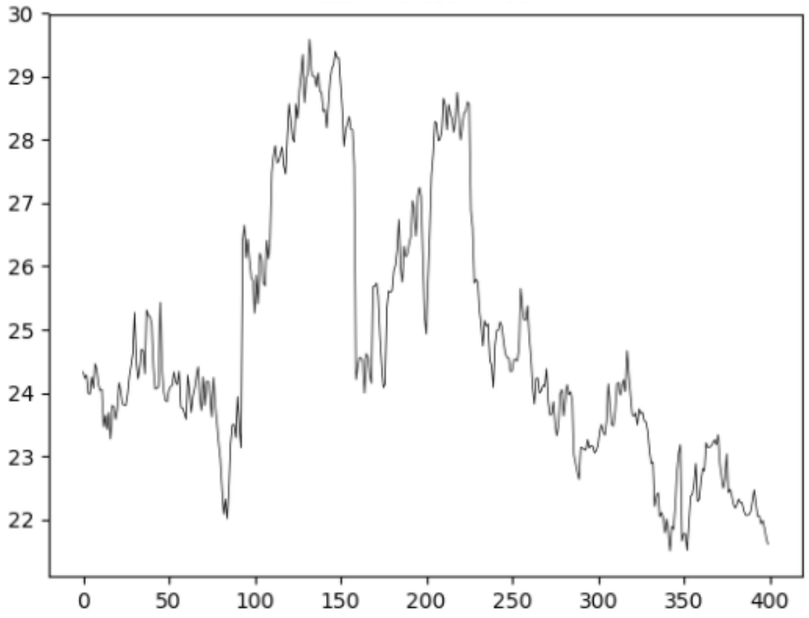
SEASONALITY

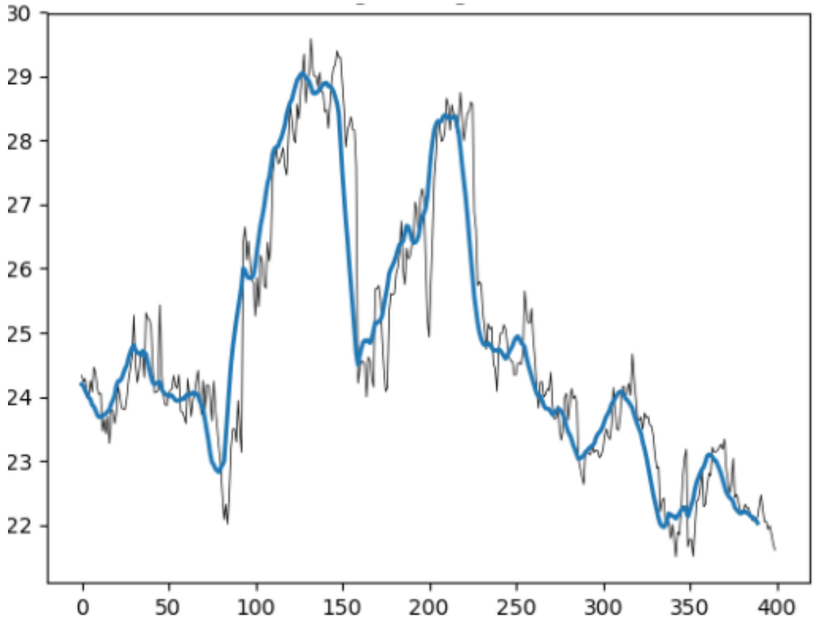
Seasonality is the presence of variations that occur at specific regular intervals. Seasonal fluctuations in a time series can be contrasted with cyclical patterns.



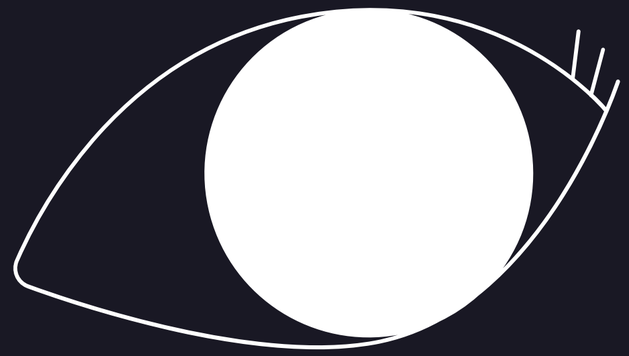
NOISE

A time series is white noise if the variables are independent and identically distributed with a mean of zero. This means that all variables have the same variance zero.



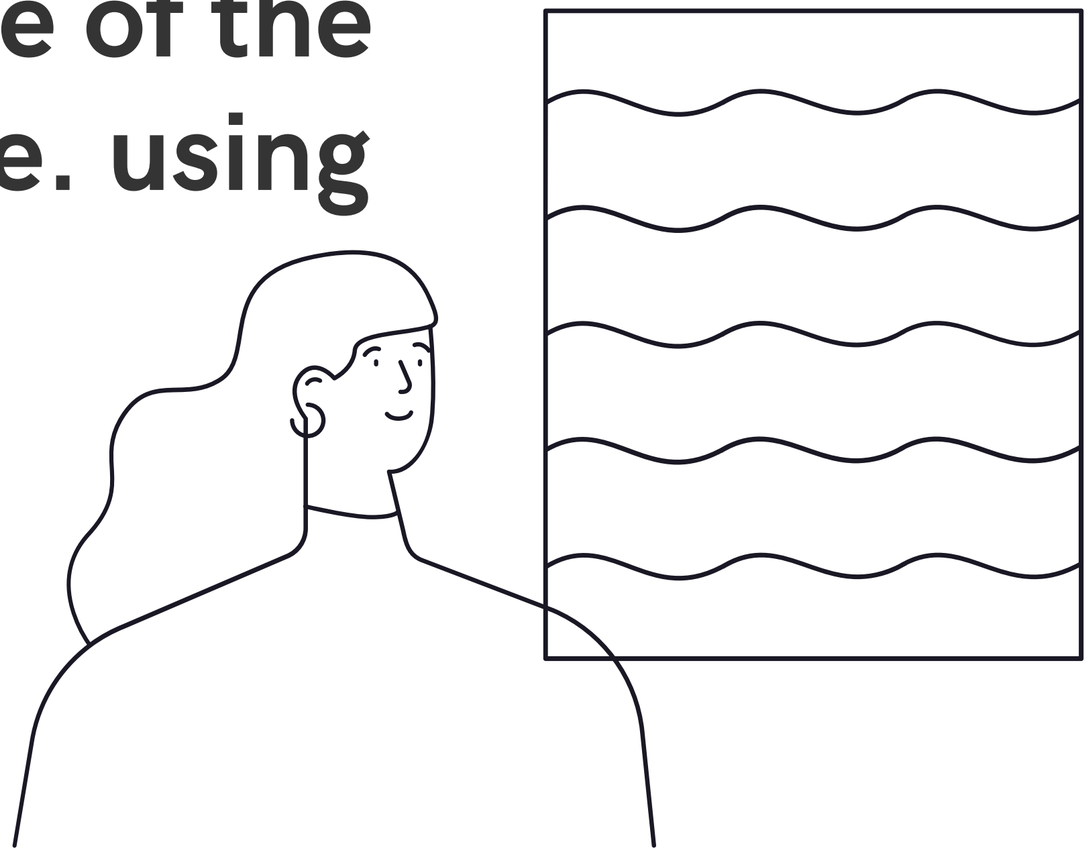


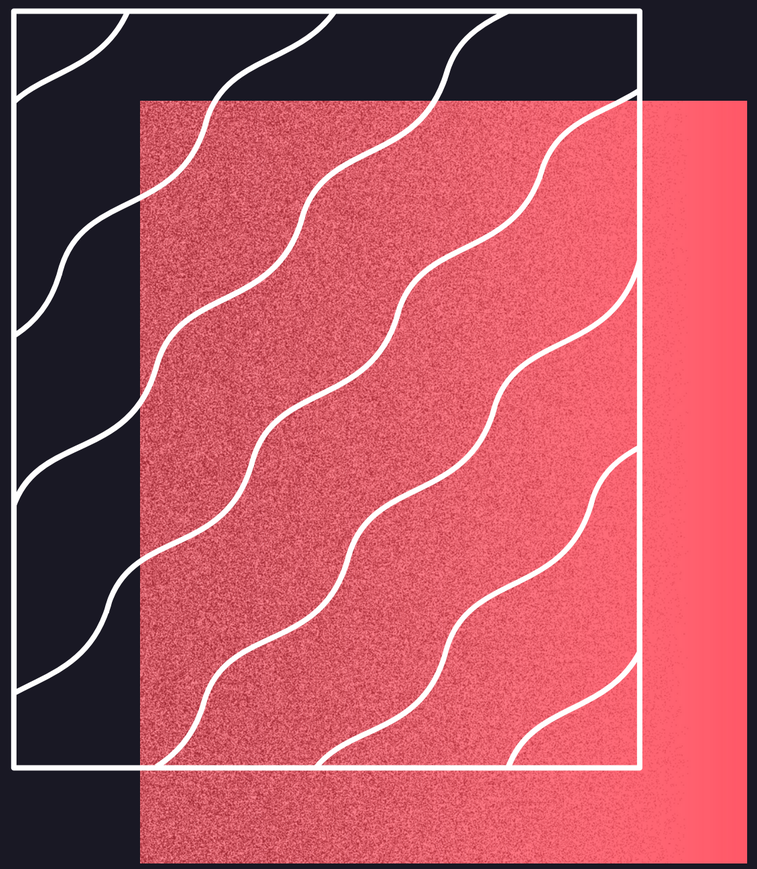
06



PREDICTION TECHNIQUES

There are a lot of ways to handle Time Series Forecasting, but we are gonna see one of the most efficient method i.e. using Deep Neural Nets.



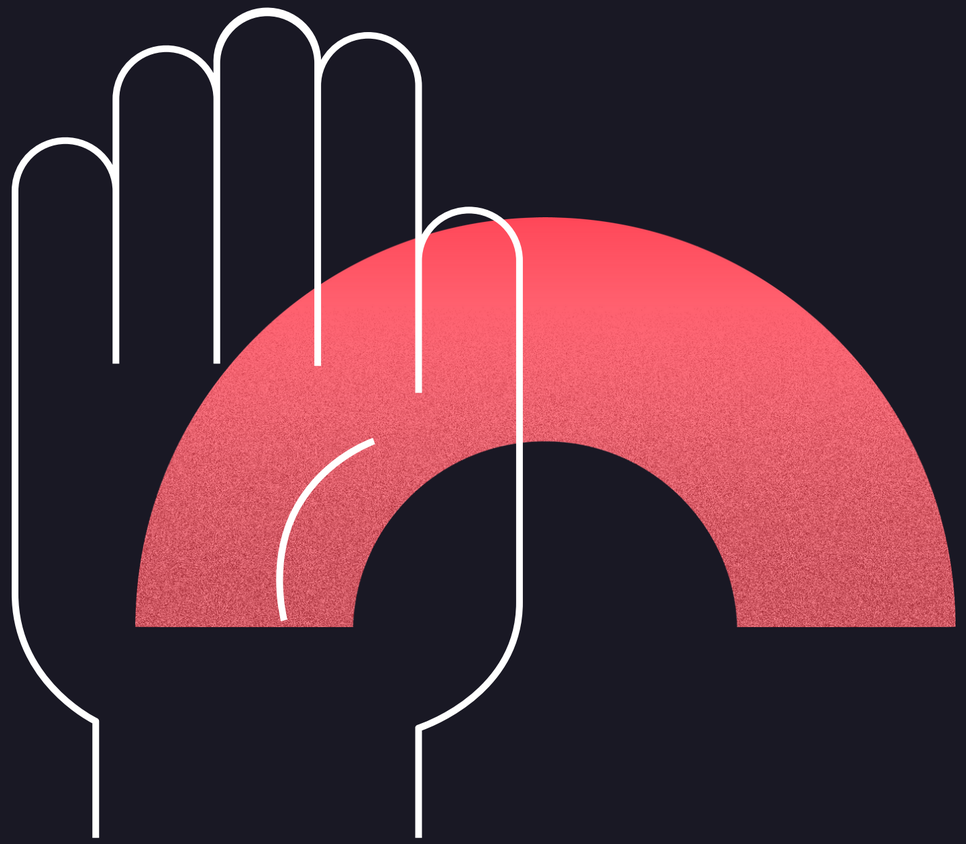


DEEP NEURAL NETS FOR TIME SERIES FORCASTING

Even there are a lot of Neural Network technique for the same, these includes:

- Artificial Neural Network (ANN)
- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
  - SIMPLE RNN
  - GRUs
  - LSTMs

WE WILL BE COVERING ALL THESE TOPICS IN DETAILS IN THE FOUR MONTH TRAINING, BUT FOR TODAY WE ARE GONNA SEE SIMPLE RNN FOR THE PREDICTION.



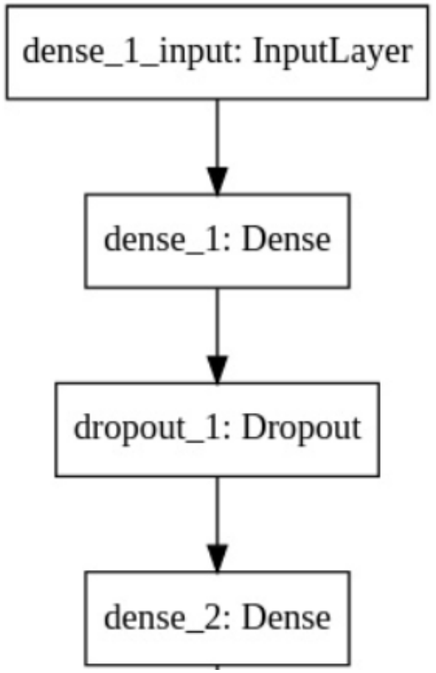
OUR MODEL

```
model = Sequential()
model.add(SimpleRNN(128, input_shape=X_train.shape[1:]))
model.add(Dropout(0.2))
model.add(Dense(3))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mse'])
model.summary()
```

SEQUENTIAL

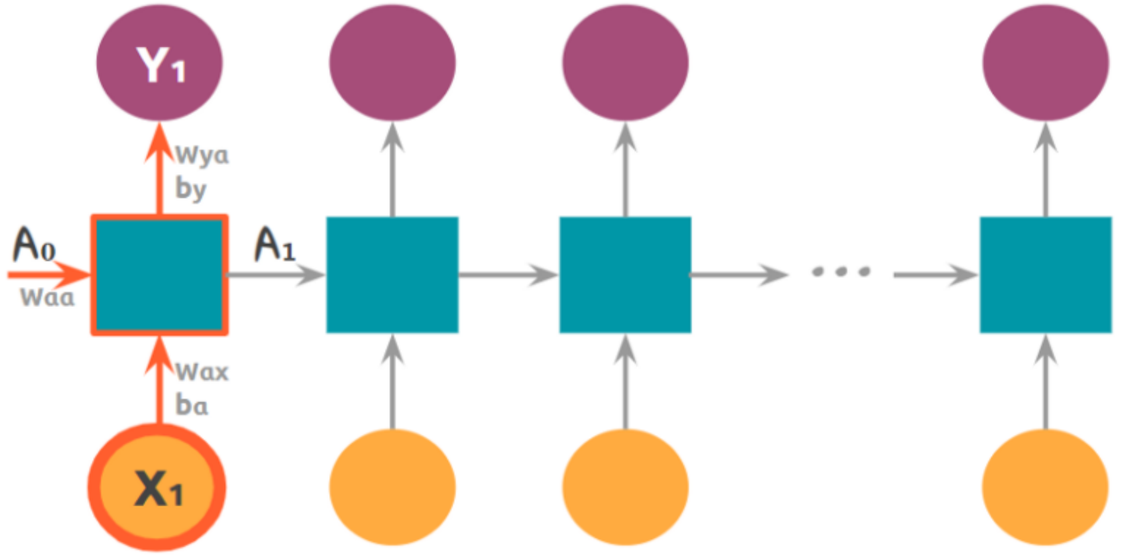
The Sequential model is a linear stack of layers. The common architecture of ConvNets is a sequential architecture. However, some architectures are not linear stacks. For example, siamese networks are two parallel neural networks with some shared layers.

A LINEAR MODEL



SIMPLE RNN

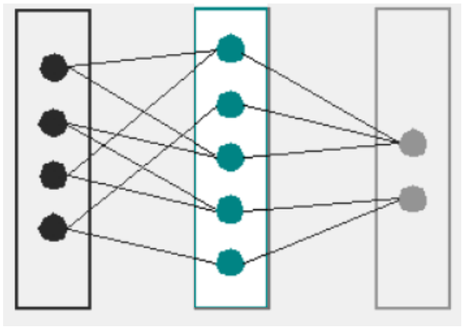
RNN has its appeal in that we can connect the data with the previous data. This means that the model starts to care about the past and what is coming next. As the recurrent units hold the past values, we can refer to this as memory.



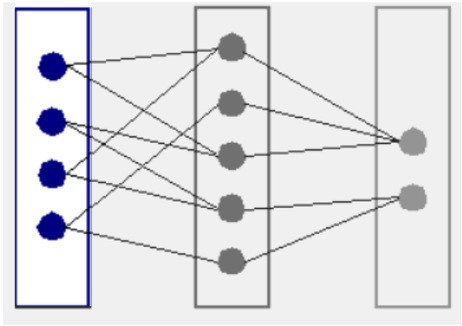
SIMPLE RNN

model.add(SimpleRNN(128, input_shape=X_train.shape[1:]))

- The very first attribute is the number of Simple RNN cells we want in our hidden RNN layer.

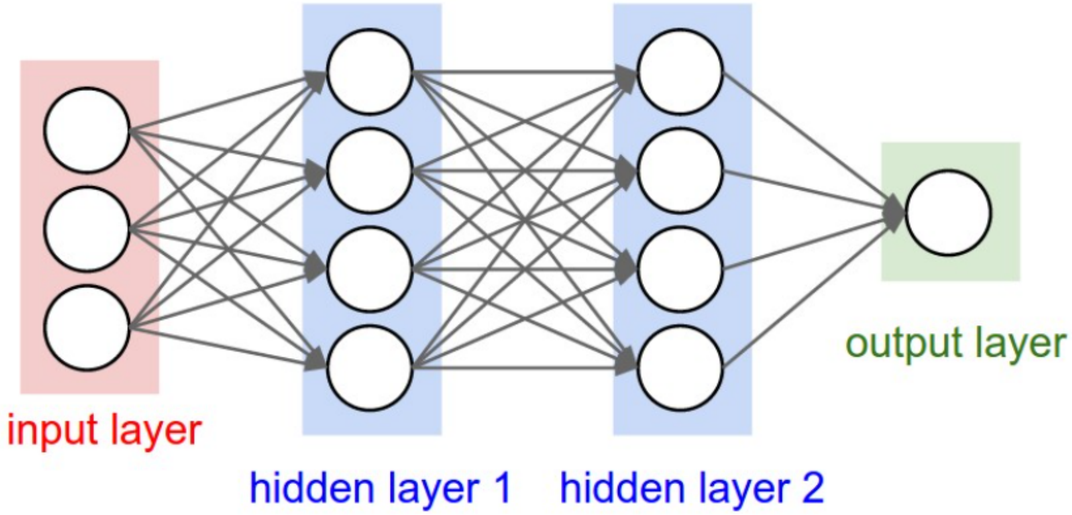


- The Second attribute is the shape of the input which we will be providing to the network.



DENSE LAYER

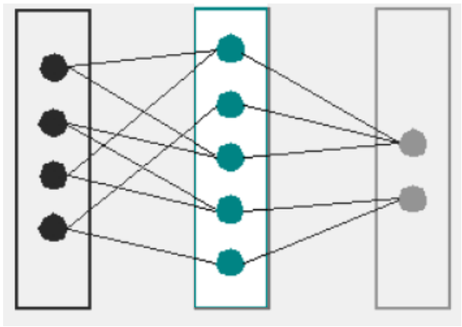
A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected.



DENSE LAYER

model.add(Dense(3))

- The one and only attribute passed is the number of dense cells we want in our dense layer

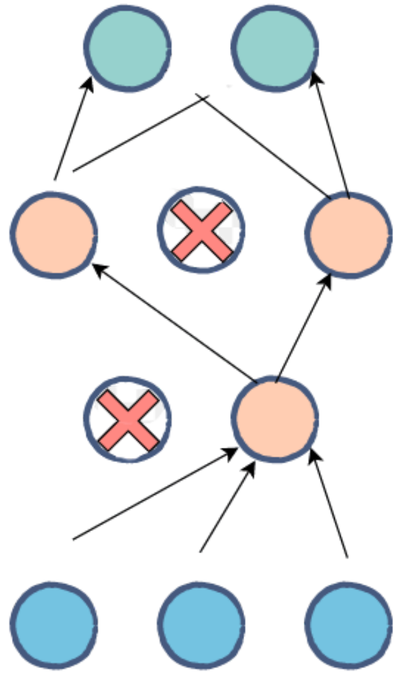


DROPOUT LAYER

model.add(Dropout (0.2))

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.

- The single parameter passed is the probability for any cell to be dropped.



COMPILE

Compile defines the loss function, the optimizer and the metrics.

```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mse'])
```

- The loss function that we have used is mean squared error.
- The optimizer used is Adam grad.
- For the metrics we are concerned with the mse error.

The exact mathematics behind these all is out of the scope of today's session.
But don't worry about that will be covering all these and many more in details during the training.

SUMMARY

model.summary()

The model summary displays the name of the model, the model type, and the model formula.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn (SimpleRNN)       (None, 128)               23680
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 3)                 387
=================================================================
```

THANKS

SHAPEAI