Welcome to Session 01

This is a Google Colaboratory (https://colab.research.google.com/notebooks/welcome.ipynb) notebook file. Python programs are run directly in the browser—a great way to learn and use TensorFlow. To follow this tutorial, run the notebook in Google Colab by clicking the button at the top of this page.

1. In Colab, connect to a Python runtime: At the top-right of the menu bar, select *CONNECT*.
2. Run all the notebook code cells: Select *Runtime* > *Run all*.

In [0]:
```
#@title ## Mounting Gdrive

USE_G_COLAB = True #@param {type:"boolean"}

if USE_G_COLAB:
    from google.colab import drive


    drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

In [0]:
```
#@title ## Project Root

root_dir = ''

if USE_G_COLAB:
    root_dir = '/content/drive/My Drive/workshops/2019_07_21/sessions_01/' #@param {type:"string"}
```

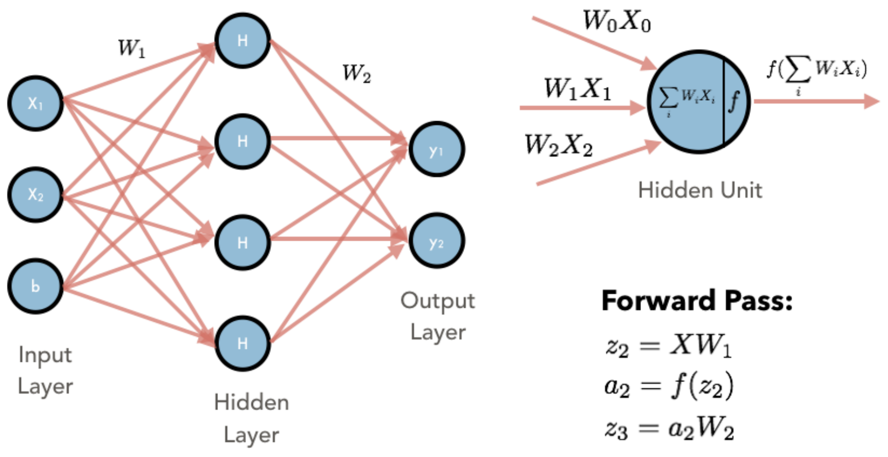
In [0]:
```
#@title ## Installing requried packages

#@markdown ---
#@markdown - [TensorFlow 2.0-beta](https://www.tensorflow.org/install/gpu)
#@markdown - [Watermark](https://github.com/rasbt/watermark)
!pip install -q tensorflow-gpu==2.0.0-beta1
!pip install -qU watermark
```

```
     |████████████████████████████████| 348.9MB 46kB/s
     |████████████████████████████████| 501kB 56.4MB/s
     |████████████████████████████████| 3.1MB 42.6MB/s
```

In [0]:
```
#@title ## Custom Matplotlib Style
mpl_style = "https://gist.githubusercontent.com/m3hrdadfi/af8aca01094afb7d3e5b46de9ad8d509/raw/871ec5
d721a3b438c3c896718ea4aafc91ea9744/gadfly.mplstyle" #@param {type:"string"}

!wget -q $mpl_style -O /root/.config/matplotlib/matplotlibrc
```

In [0]:
```
#@title ## General Paramas

#@markdown > A random seed is a number used to initialize a pseudorandom number generator. For a seed
to be used in a pseudorandom number generator, it does not need to be random
RANDOM_SEED = 141 #@param {type:"integer"}
```

Overview



$z_2 = XW_1$

$a_2 = f(z_2)$

$z_3 = a_2W_2$

$\hat{y} = softmax(z_3)$ # classification

where:

- X = inputs | $\in \mathbb{R}^{NXD}$ (D is the number of features)
- W_1 = 1st layer weights | $\in \mathbb{R}^{DXH}$ (H is the number of hidden units in layer 1)
- z_2 = outputs from first layer's weights $\in \mathbb{R}^{NXH}$
- f = non-linear activation function
- a_2 = activation applied first layer's outputs | $\in \mathbb{R}^{NXH}$
- W_2 = 2nd layer weights | $\in \mathbb{R}^{HXC}$ (C is the number of classes)
- $\hat{y}$ = prediction | $\in \mathbb{R}^{NXC}$ (N is the number of samples)

This is a simple two-layer MLP.

- **Objective:** Predict the probability of class y given the inputs X. Non-linearity is introduced to model the complex, non-linear data.
- **Advantages:**
  - Can model non-linear patterns in the data really well.

- **Disadvantages:**
  - Overfits easily.
  - Computationally intensive as network increases in size.
  - Not easily interpretable.
- **Miscellaneous:** Future neural network architectures that we'll see use the MLP as a modular unit for feed forward operations (affine transformation (XW) followed by a non-linear operation).

7 Common Nonlinear Activation Functions and How to Choose an Activation Function

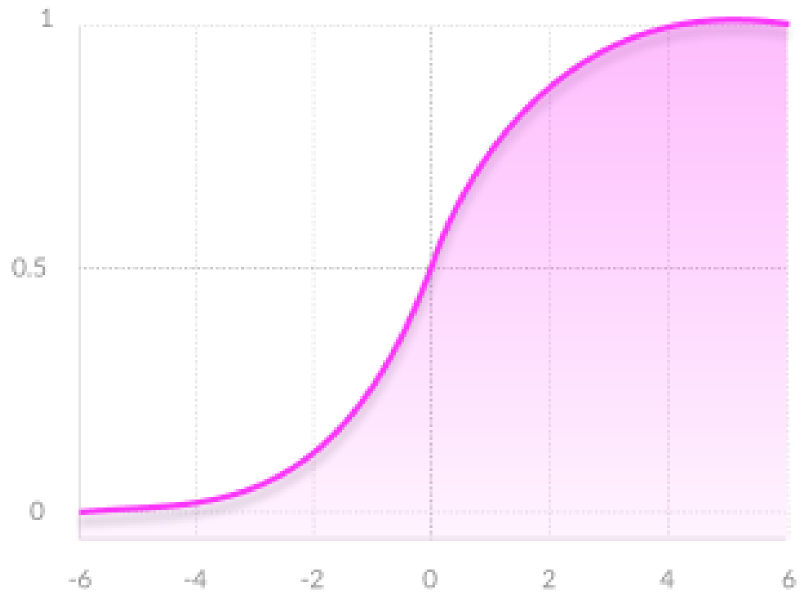
SIGMOID / LOGISTIC

ADVANTAGES

- Smooth gradient, preventing “jumps” in output values.
- Output values bound between 0 and 1, normalizing the output of each neuron.
- Clear predictions—For X above 2 or below -2, tends to bring the Y value (the prediction) to the edge of the curve, very close to 1 or 0. This enables clear predictions.

DISADVANTAGES

- Vanishing gradient—for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.
- Outputs not zero centered.
- Computationally expensive



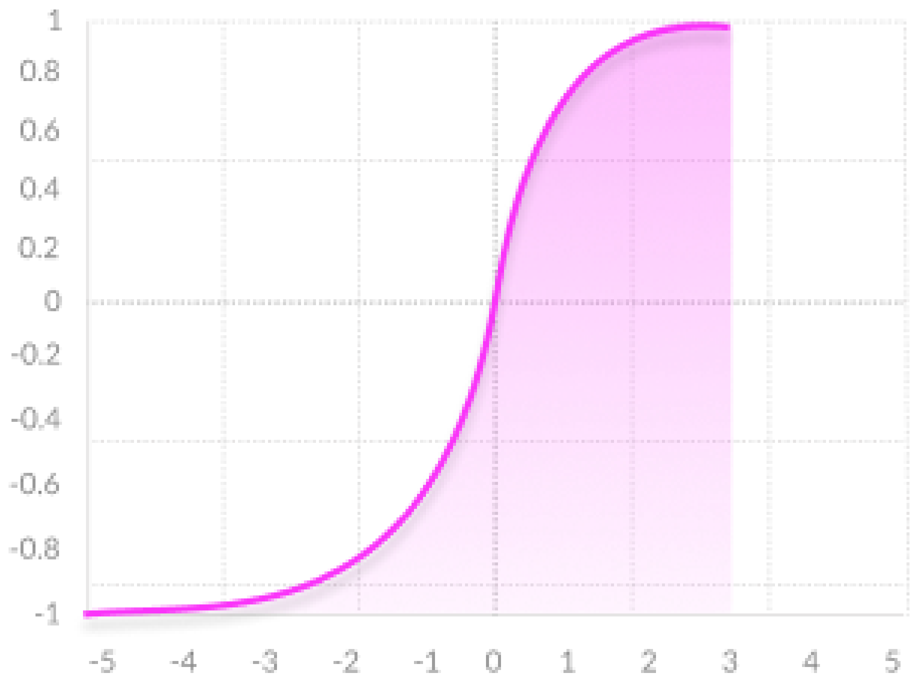
TANH / HYPERBOLIC TANGENT

ADVANTAGES

- Zero centered—making it easier to model inputs that have strongly negative, neutral, and strongly positive values.
- Otherwise like the Sigmoid function.

DISADVANTAGES

- Like the Sigmoid function



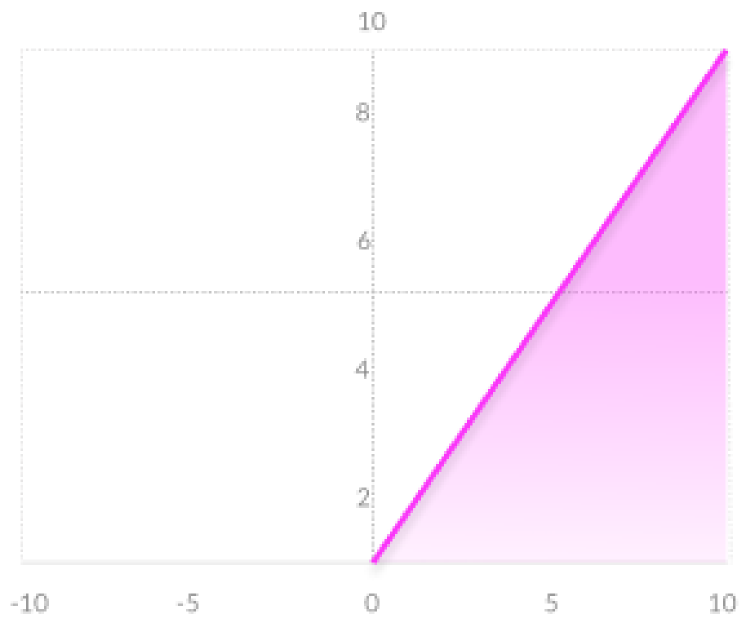
RELU (RECTIFIED LINEAR UNIT)

ADVANTAGES

- Computationally efficient—allows the network to converge very quickly
- Non-linear—although it looks like a linear function, ReLU has a derivative function and allows for backpropagation.

DISADVANTAGES

- The Dying ReLU problem—when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn.
- Like the Sigmoid function

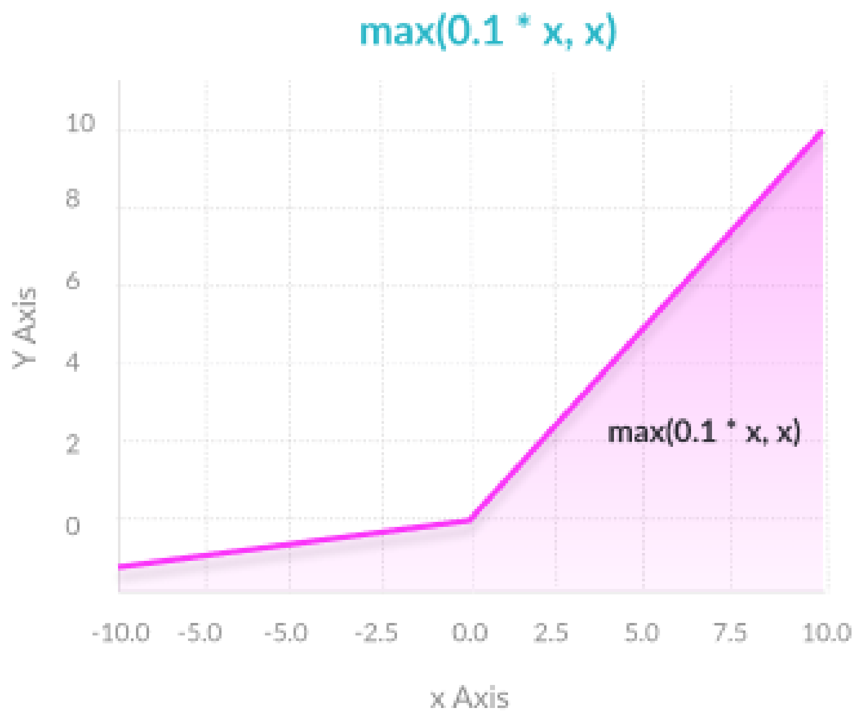


LEAKY RELU

ADVANTAGES

- Prevents dying ReLU problem—this variation of ReLU has a small positive slope in the negative area, so it does enable backpropagation, even for negative input values Otherwise like ReLU

DISADVANTAGES Results not consistent—leaky ReLU does not provide consistent predictions for negative input values.



PARAMETRIC RELU

ADVANTAGES

- Allows the negative slope to be learned—unlike leaky ReLU, this function provides the slope of the negative part of the function as an argument. It is, therefore, possible to perform backpropagation and learn the most appropriate value of α. Otherwise like ReLU

DISADVANTAGES May perform differently for different problems.

$f(x) = max(\alpha x, x)$

SOFTMAX

ADVANTAGES

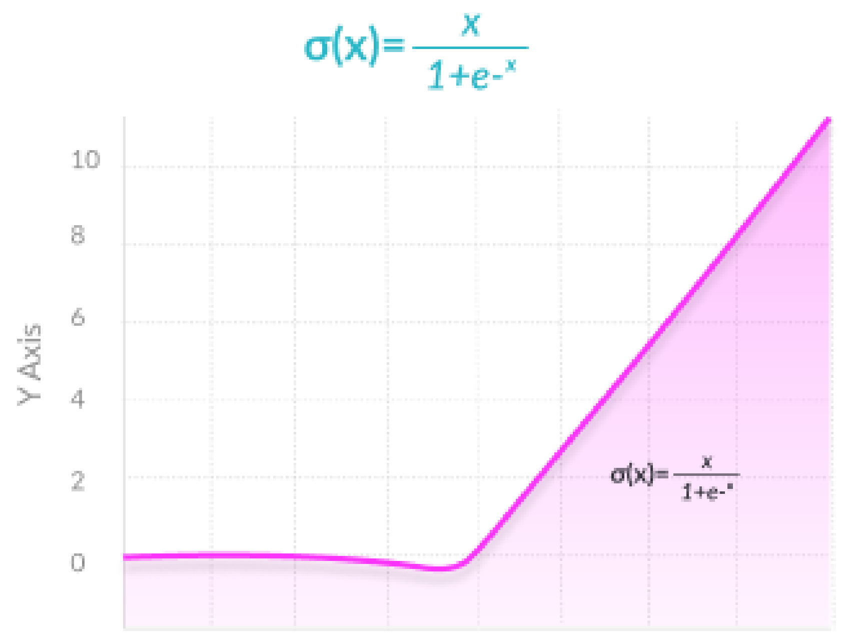
- Able to handle multiple classes only one class in other activation functions—normalizes the outputs for each class between 0 and 1, and divides by their sum, giving the probability of the input value being in a specific class.

- Useful for output neurons—typically Softmax is used only for the output layer, for neural networks that need to classify inputs into multiple categories.

$$\frac{e^{z_j}}{\sum_{k=1}^{c} e^{z_k}}$$

SWISH

Swish is a new, self-gated activation function discovered by researchers at Google. According to their paper, it performs better than ReLU with a similar level of computational efficiency. In experiments on ImageNet with identical models running ReLU and Swish, the new function achieved top -1 classification accuracy 0.6-0.9% higher.



Training

Steps:

1. Randomly initialize the model's weights W (we'll cover more effective initalization strategies in future lessons).
2. Feed inputs X into the model to do the forward pass and receive the probabilities.
3. Compare the predictions $\hat{y}$ (ex. [0.3, 0.3, 0.4]]) with the actual target values y (ex. class 2 would look like [0, 0, 1]) with the objective (cost) function to determine loss J. A common objective function for classification tasks is cross-entropy loss.
   - $z_2 = XW_1$
   - $a_2 = max(0, z_2)$ # ReLU activation
   - $z_3 = a_2W_2$
   - $\hat{y} = softmax(z_3)$
   - $J(\theta) = - \sum_i y_i ln(\hat{y_i})$
4. Calculate the gradient of loss $J(\theta)$ w.r.t to the model weights.
   - $\frac{\partial J}{\partial W_{2j}} = a_2\hat{y}, \frac{\partial J}{\partial W_{2y}} = a_2(\hat{y} - 1)$
   - $\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial W_1} = W_2(\partial scores)(\partial ReLU)X$
5. Apply backpropagation to update the weights W using gradient descent. The updates will penalize the probabiltiy for the incorrect classes (j) and encourage a higher probability for the correct class (y).
   - $W_i = W_i - \alpha\frac{\partial J}{\partial W_i}$
6. Repeat steps 2 - 4 until model performs well.

Terminologies

Neural Network Learning as Optimization

A deep learning neural network learns to map a set of inputs to a set of outputs from training data.

We cannot calculate the perfect weights for a neural network; there are too many unknowns. Instead, the problem of learning is cast as a search or optimization problem and an algorithm is used to navigate the space of possible sets of weights the model may use in order to make good or good enough predictions.

Typically, a neural network model is trained using the stochastic gradient descent optimization algorithm and weights are updated using the backpropagation of error algorithm.

The "gradient" in gradient descent refers to an error gradient. The model with a given set of weights is used to make predictions and the error for those predictions is calculated.

The gradient descent algorithm seeks to change the weights so that the next evaluation reduces the error, meaning the optimization algorithm is navigating down the gradient (or slope) of error.

Now that we know that training neural nets solves an optimization problem, we can look at how the error of a given set of weights is calculated.

What Is a Loss Function and Loss?

In the context of an optimization algorithm, the function used to evaluate a candidate solution (i.e. a set of weights) is referred to as the objective function.

We may seek to maximize or minimize the objective function, meaning that we are searching for a candidate solution that has the highest or lowest score respectively.

Typically, with neural networks, we seek to minimize the error. As such, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply "loss." The cost or loss function has an important job in that it must faithfully distill all aspects of the model down into a single number in such a way that improvements in that number are a sign of a better model. We will review best practice or default values for each problem type with regard to the output layer and loss function.

Regression Problem

A problem where you predict a real-value quantity.

Output Layer Configuration: One node with a linear activation unit. Loss Function: Mean Squared Error (MSE).

Binary Classification Problem

A problem where you classify an example as belonging to one of two classes.

The problem is framed as predicting the likelihood of an example belonging to class one, e.g. the class that you assign the integer value 1, whereas the other class is assigned the value 0.

Output Layer Configuration: One node with a sigmoid activation unit. Loss Function: Cross-Entropy, also referred to as Logarithmic loss.

Multi-Class Classification Problem

A problem where you classify an example as belonging to one of more than two classes.

The problem is framed as predicting the likelihood of an example belonging to each class.

Output Layer Configuration: One node for each class using the softmax activation function. Loss Function: Cross-Entropy, also referred to as Logarithmic loss.

How to Implement Loss Functions

In order to make the loss functions concrete, this section explains how each of the main types of loss function works.

Mean Squared Error Loss

Mean Squared Error loss, or MSE for short, is calculated as the average of the squared differences between the predicted and actual values.

The result is always positive regardless of the sign of the predicted and actual values and a perfect value is 0.0. The loss value is minimized, although it can be used in a maximization optimization process by making the score negative.

Cross-Entropy Loss (or Log Loss)

Cross-entropy loss is often simply referred to as “cross-entropy,” “logarithmic loss,” “logistic loss,” or “log loss” for short.

Each predicted probability is compared to the actual class output value (0 or 1) and a score is calculated that penalizes the probability based on the distance from the expected value. The penalty is logarithmic, offering a small score for small differences (0.1 or 0.2) and enormous score for a large difference (0.9 or 1.0).

Cross-entropy loss is minimized, where smaller values represent a better model than larger values. A model that predicts perfect probabilities has a cross entropy or log loss of 0.0.

Cross-entropy for a binary or two class prediction problem is actually calculated as the average cross entropy across all examples.

An overview of gradient descent optimization algorithms

Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. At the same time, every state-of-the-art Deep Learning library contains implementations of various algorithms to optimize gradient descent. These algorithms, however, are often used as black-box optimizers, as practical explanations of their strengths and weaknesses are hard to come by.

Gradient descent variants

There are three variants of gradient descent, which differ in how much data we use to compute the gradient of the objective function. Depending on the amount of data, we make a trade-off between the accuracy of the parameter update and the time it takes to perform an update.

Batch gradient descent

Vanilla gradient descent, aka batch gradient descent, computes the gradient of the cost function w.r.t. to the parameters θ for the entire training dataset:

$\theta = \theta - \eta \nabla_\theta J(\theta)$
We then update our parameters in the opposite direction of the gradients with the learning rate determining how big of an update we perform. Batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces. As we need to calculate the gradients for the whole dataset to perform just one update, batch gradient descent can be very slow and is intractable for datasets that don't fit in memory. Batch gradient descent also doesn't allow us to update our model online, i.e. with new examples on-the-fly. ## Stochastic gradient descent Stochastic gradient descent (SGD) in contrast performs a parameter update for each training example x(i) and label y(i):

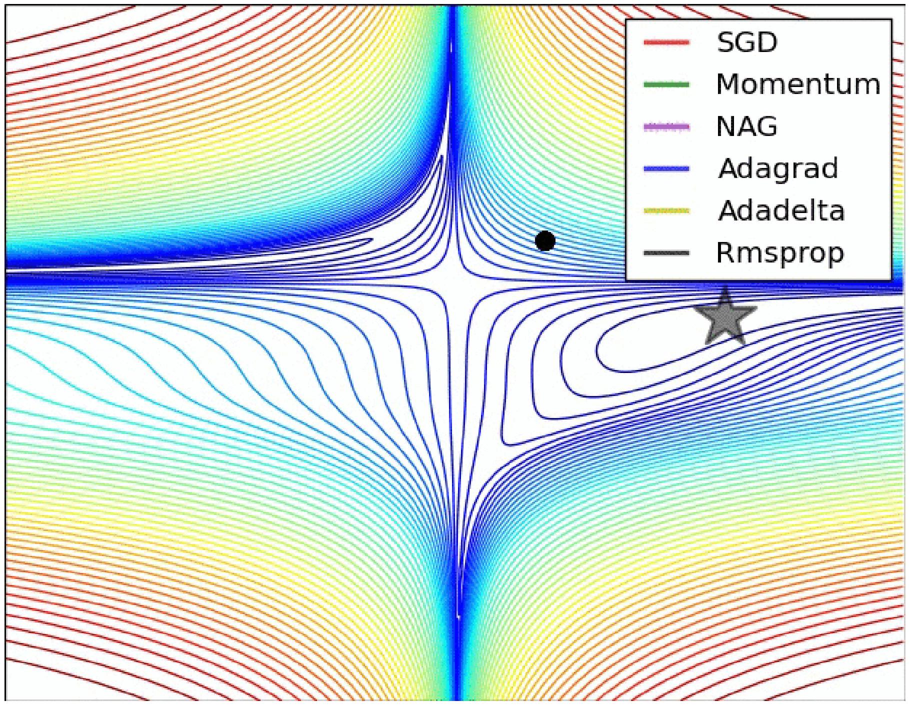
$\theta = \theta - \eta \nabla_\theta J(\theta; x^i; y^i)$
Batch gradient descent performs redundant computations for large datasets, as it recomputes gradients for similar examples before each parameter update. SGD does away with this redundancy by performing one update at a time. It is therefore usually much faster and can also be used to learn online. While batch gradient descent converges to the minimum of the basin the parameters are placed in, SGD's fluctuation, on the one hand, enables it to jump to new and potentially better local minima. On the other hand, this ultimately complicates convergence to the exact minimum, as SGD will keep overshooting. However, it has been shown that when we slowly decrease the learning rate, SGD shows the same convergence behaviour as batch gradient descent, almost certainly converging to a local or the global minimum for non-convex and convex optimization respectively. ## Mini-batch gradient descent Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n training examples:

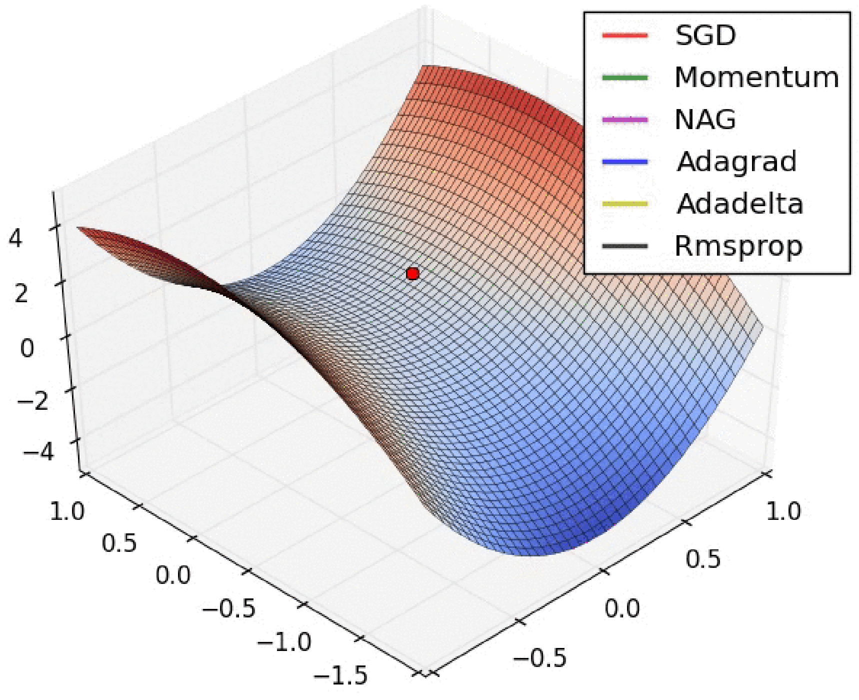
$\theta = \theta - \eta \nabla_\theta J(\theta; x^{i:i+n}; y^{i:i+n})$

This way, it a) reduces the variance of the parameter updates, which can lead to more stable convergence; and b) can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient. Common mini-batch sizes range between 50 and 256, but can vary for different applications. Mini-batch gradient descent is typically the algorithm of choice when training a neural network and the term SGD usually is employed also when mini-batches are used.

Gradient descent optimization algorithms

- Momentum
- Nesterov accelerated gradient (NAG)
- Adagrad
- Adadelta
- RMSprop
- Adam
- AdaMax
- Nadam
- AMSGrad





Which optimizer to use?

So, which optimizer should you now use? If your input data is sparse, then you likely achieve the best results using one of the adaptive learning-rate methods. An additional benefit is that you won't need to tune the learning rate but likely achieve the best results with the default value.

In summary, RMSprop is an extension of Adagrad that deals with its radically diminishing learning rates. It is identical to Adadelta, except that Adadelta uses the RMS of parameter updates in the numinator update rule. Adam, finally, adds bias-correction and momentum to RMSprop. Insofar, RMSprop, Adadelta, and Adam are very similar algorithms that do well in similar circumstances. It is shown that its bias-correction helps Adam slightly outperform RMSprop towards the end of optimization as gradients become sparser. Insofar, Adam might be the best overall choice.

Hyperparameters

Model optimization is one of the toughest challenges in the implementation of machine learning solutions. Entire branches of machine learning and deep learning theory have been dedicated to the optimization of models. Typically, we think about model optimization as a process of regularly modifying the code of the model in order to minimize the testing error. However, deep learning optimization often entails fine tuning elements that live outside the model but that can heavily influence its behavior. Deep learning often refers to those hidden elements as hyperparameters as they are one of the most critical components of any machine learning application.

Some Examples of Hyperparameters

The number and diversity of hyperparameters in machine learning algorithms is very specific to each model. However, there some classic hyperparameters that we should always keep our eyes on and that should help you think about this aspect of machine learning solutions: · Learning Rate: The mother of all hyperparameters, the learning rate quantifies the learning progress of a model in a way that can be used to optimize its capacity. · Number of Hidden Units: A classic hyperparameter in deep learning algorithms, the number of hidden units is key to regulate the representational capacity of a model. · Convolution Kernel Width: In convolutional Neural Networks(CNNs), the Kernel Width influences the number of parameters in a model which, in turns, influences its capacity.

Metrics to Evaluate your Machine Learning Algorithm

Evaluating your machine learning algorithm is an essential part of any project. Your model may give you satisfying results when evaluated using a metric say accuracy_score but may give poor results when evaluated against other metrics such as logarithmic_loss or any other such metric. Most of the times we use classification accuracy to measure the performance of our model, however it is not enough to truly judge our model. In this post, we will cover different types of evaluation metrics available.

Classification Accuracy

Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

$Accuracy = \frac{Number of Correct predictions}{Total Number of Predictions made}$

It works well only if there are equal number of samples belonging to each class. For example, consider that there are 98% samples of class A and 2% samples of class B in our training set. Then our model can easily get 98% training accuracy by simply predicting every training sample belonging to class A. When the same model is tested on a test set with 60% samples of class A and 40% samples of class B, then the test accuracy would drop down to 60%. Classification Accuracy is great, but gives us the false sense of achieving high accuracy. The real problem arises, when the cost of misclassification of the minor class samples are very high. If we deal with a rare but fatal disease, the cost of failing to diagnose the disease of a sick person is much higher than the cost of sending a healthy person to more tests.

Confusion Matrix

Confusion Matrix as the name suggests gives us a matrix as output and describes the complete performance of the model. Lets assume we have a binary classification problem. We have some samples belonging to two classes : YES or NO. Also, we have our own classifier which predicts a class for a given input sample. On testing our model on 165 samples ,we get the following result.

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i \cdot log_e(\hat{y_i}) + (1 - y_i) \cdot log_e(1 - \hat{y_i})]$$

where, y_ij, indicates whether sample i belongs to class j or not p_ij, indicates the probability of sample i belonging to class j Log Loss has no upper bound and it exists on the range Log Loss nearer to 0 indicates higher accuracy, whereas if the Log Loss is away from 0 then it indicates lower accuracy. In general, minimising Log Loss gives greater accuracy for the classifier.

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

There are 4 important terms :

- True Positives : The cases in which we predicted YES and the actual output was also YES.
- True Negatives : The cases in which we predicted NO and the actual output was NO.
- False Positives : The cases in which we predicted YES and the actual output was NO.
- False Negatives : The cases in which we predicted NO and the actual output was YES. Accuracy for the matrix can be calculated by taking average of the values lying across the “main diagonal” i.e
  $$Accuracy = \frac{TruePositive+TrueNegative}{TotalNumberofSamples}$$
  $$Accuracy = \frac{100+50}{165} = 0.91$$

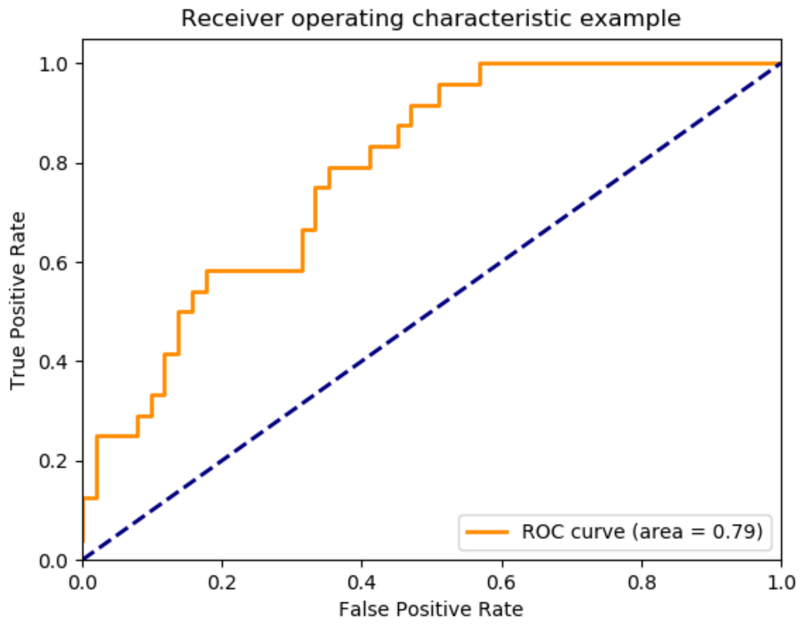
Confusion Matrix forms the basis for the other types of metrics.

Area Under Curve

Area Under Curve(AUC) is one of the most widely used metrics for evaluation. It is used for binary classification problem. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. Before defining AUC, let us understand two basic terms :

- True Positive Rate (Sensitivity) : True Positive Rate is defined as TP/ (FN+TP). True Positive Rate corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points.
  $$TruePositiveRate = \frac{TruePositive}{FalseNegative+TruePositive}$$
- False Positive Rate (Specificity) : False Positive Rate is defined as FP / (FP+TN). False Positive Rate corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points.
  $$FalsePositiveRate = \frac{FalsePositive}{TrueNegative+FalsePositive}$$

False Positive Rate and True Positive Rate both have values in the range [0, 1]. FPR and TPR bot hare computed at threshold values such as (0.00, 0.02, 0.04, …., 1.00) and a graph is drawn. AUC is the area under the curve of plot False Positive Rate vs True Positive Rate at different points in [0, 1].



As evident, AUC has a range of [0, 1]. The greater the value, the better is the performance of our model.

F1 Score

F1 Score is used to measure a test’s accuracy F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances). High precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model. Mathematically, it can be expressed as :

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

- Precision : It is the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Precision = 2 * \frac{TruePositive}{TruePositive + FalsePositive}$$

- Recall : It is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

$$Recall = 2 * \frac{TruePositive}{TruePositive + FalsePositive}$$

Preparation

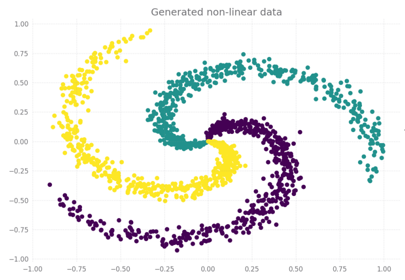
In [0]:
```
#@title ## Import requried packages
from __future__ import absolute_import, division, print_function, unicode_literals

import tensorflow as tf

import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

import matplotlib as mpl
import matplotlib.pyplot as plt

import itertools
from datetime import datetime
from tqdm import tqdm
import os

%matplotlib inline
mpl.rc_file(mpl.matplotlib_fname())
```

In [0]:
```
%reload_ext watermark
%watermark -m -n -p tensorflow,numpy,matplotlib,sklearn -g
```

```
Fri Jul 19 2019

tensorflow 2.0.0-beta1
numpy 1.16.4
matplotlib 3.0.3
sklearn 0.21.2

compiler   : GCC 8.0.1 20180414 (experimental) [trunk revision 259383
system     : Linux
release    : 4.14.79+
machine    : x86_64
processor  : x86_64
CPU cores  : 2
interpreter: 64bit
Git hash   :
```

In [0]:
```
#@title ## Random seed

np.random.seed(RANDOM_SEED)
```

Data



We're going to first generate some non-linear data for a classification task.

A spiral: is a curve which emanates from a point, moving farther away as it revolves around the point.

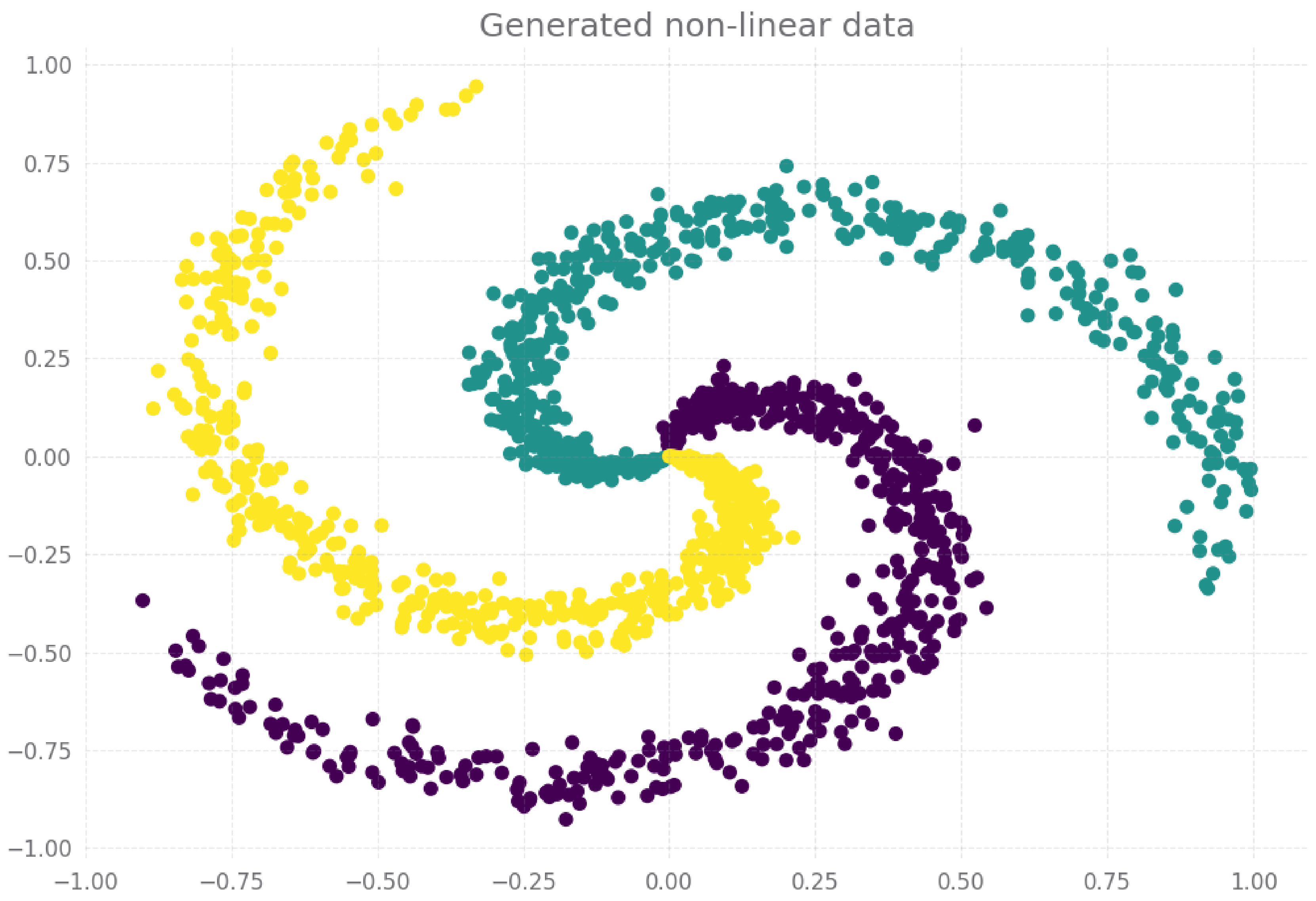
$$Spiral = \begin{cases} X_{1\theta} = r_\theta \cos(\theta) \\ X_{2\theta} = r_\theta \sin(\theta) \end{cases}$$

In [0]:
```
#@title ## Generate non-linear data [Spiral](https://en.wikipedia.org/wiki/Spiral)

def generate_data(num_samples, dimensions, num_classes):
    """ Generate non-linear dataset.

    Args:
        num_samples (int): number of samples which we want to produce.
        dimensions (int): the dimension of the data which we plan to produce ex. 2d.
        num_classes (int): number of classes which the samples are going to generate ex. 2#.

    Examples:
        >>> x, y = generate_data(2, 2, 2)
        (array([[ 0.        ,  0.        ],
        [-0.712528  , -0.70164368],
        [-0.        , -0.        ],
        [ 0.90006129, -0.43576333]]), array([0, 0, 1, 1], dtype=uint8))

    Returns:
        x (ndarray): a numpy array in a shape like this (num_samples, dimensions).
        y (ndarray): a numpy array in a shape like this (num_samples, num_classes).
    """

    x_origin = np.zeros((num_samples * num_classes, dimensions))
    y = np.zeros(num_samples * num_classes, dtype='uint8')

    for i in tqdm(range(num_classes), position=0):
        idx = range(num_samples * i, num_samples * (i + 1))
        radius = np.linspace(0.0, 1, num_samples)
        theta = np.linspace(i * 4, (i + 1) * 4, num_samples) + np.random.randn(num_samples) * 0.2

        x_origin[idx] = np.c_[radius * np.sin(theta), radius * np.cos(theta)]
        y[idx] = i

    x = np.hstack([x_origin])

    return x, y
```

In [0]:
```
#@title ## Generate x, y

num_samples = 500 #@param {type:"integer"}
dimensions = 2 #@param {type:"integer"}
num_classes = 3 #@param {type:"integer"}

x, y = generate_data(num_samples, dimensions, num_classes)

print()
print('x: %s' % str(x.shape))
print('y: %s' % str(y.shape))
```

```
100%|██████████| 3/3 [00:00<00:00, 681.71it/s]

x: (1500, 2)
y: (1500,)
```

In [0]:
```
#@title ## Visualize data

xfig = 12.0 #@param {type:"number"}
yfig = 8.0 #@param {type:"number"}

plt.figure(figsize=(xfig, yfig))
plt.title('Generated non-linear data')
plt.scatter(x[:, 0], x[:, 1], c=y)
plt.show()
```


Generated non-linear data
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

One-hot Encoding

Consider an array of 5 labels out of a set of 3 classes {0, 1, 2}:

```
> labels
array([0, 2, 1, 2, 0])
```

`to_categorical` converts this into a matrix with as many columns as there are classes. The number of rows stays the same.

```
> to_categorical(labels)
array([[ 1.,  0.,  0.],
       [ 0.,  0.,  1.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.],
       [ 1.,  0.,  0.]], dtype=float32)
```

---

to_categorical

```
tf.keras.utils.to_categorical(y, num_classes=None, dtype='float32')
```

Converts a class vector (integers) to binary class matrix.

Arguments:

- **`y`** : class vector to be converted into a matrix (integers from 0 to num_classes).
- **`num_classes`** : total number of classes.
- **`dtype`** : The data type expected by the input, as a string (float32, float64, int32...)

```
In [0]: #@title ## Preprocessing

        test_size = 0.33 #@param {type:"number"}
        shuffling = True #@param {type:"boolean"}

        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=RANDOM_SE
        ED, shuffle=shuffling)

        print('Train: [x]=> %s,\t [y]=> %s' % (x_train.shape, y_train.shape))
        print('Test:  [x]=> %s,\t [y]=> %s' % (x_test.shape, y_test.shape))


        y_train_c = tf.keras.utils.to_categorical(y_train, num_classes)
        y_test_c = tf.keras.utils.to_categorical(y_test, num_classes)

        print('Train: [x]=> %s,\t [y]=> %s' % (x_train.shape, y_train_c.shape))
        print('Test:  [x]=> %s,\t [y]=> %s' % (x_test.shape, y_test_c.shape))
```

```
Train: [x]=> (1005, 2),   [y]=> (1005,)
Test:  [x]=> (495, 2),    [y]=> (495,)
Train: [x]=> (1005, 2),   [y]=> (1005, 3)
Test:  [x]=> (495, 2),    [y]=> (495, 3)
```

Linear Model

Before we get to our neural network, we're going to implement a linear model (logistic regression). We want to see why linear models won't suffice for our dataset.

In [0]:
```
#@title ## Build linear model [Logistic Model](https://en.wikipedia.org/wiki/Multinomial_logistic_reg
ression)

def build_linear(n_units, n_features, compiling=True, opt='adam', loss='categorical_crossentropy', me
trics=None):
    """ Build a linear model [Logistic Model].

    Args:
        n_units (int): dimensionality of the output space.
        n_features (int): indicates that the expected input will be batches of n_features-dimensional
vectors.
        compiling (boolean): whether the model complied or not.
        opt (str, keras.optimizers): optimizer instance.
        loss (str, keras.losses): objective function.
        metrics (str, keras.metrics): List of metrics to be evaluated by the model during training an
d testing.

    Returns:
        model (keras.Model): a keras model.
    """

    model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(n_features,)),
        tf.keras.layers.Dense(n_units, activation='softmax')
    ])

    if compiling:

        if metrics is None or not isinstance(metrics, list):
            metrics = ['accuracy']

        model.compile(optimizer=opt, loss=loss, metrics=metrics)

    return model
```

In [0]:
```
#@title ## Create linear model

compiling = True #@param {type:"boolean"}

opt = 'adam' #@param {type:"string"}
loss = 'categorical_crossentropy' #@param {type:"string"}
metrics = "accuracy" #@param ["accuracy"] {allow-input: true}

n_units = y_train_c.shape[1] # 3, num_classes
n_features = x_train.shape[1] # 2, dimensions


linear_model = build_linear(n_units, n_features, compiling, opt, loss, metrics)
linear_model
```

Out[0]:
```
<tensorflow.python.keras.engine.sequential.Sequential at 0x7fa4b0686940>
```

In [0]:
```
#@title ## Fit linear model.

r = linear_model.fit(x_train, y_train_c, epochs=100, verbose=1)

history_dict = r.history
history_list = list(history_dict.keys())

print()
print('history_list', history_list)
```

```
Train on 1005 samples
Epoch 1/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.8968 - accuracy: 0.5413
Epoch 2/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.8926 - accuracy: 0.5423
Epoch 3/100
1005/1005 [==============================] - 0s 64us/sample - loss: 0.8885 - accuracy: 0.5403
Epoch 4/100
1005/1005 [==============================] - 0s 64us/sample - loss: 0.8844 - accuracy: 0.5413
Epoch 5/100
1005/1005 [==============================] - 0s 65us/sample - loss: 0.8805 - accuracy: 0.5383
Epoch 6/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.8767 - accuracy: 0.5373
Epoch 7/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.8731 - accuracy: 0.5383
Epoch 8/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.8695 - accuracy: 0.5373
Epoch 9/100
1005/1005 [==============================] - 0s 71us/sample - loss: 0.8661 - accuracy: 0.5383
Epoch 10/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.8628 - accuracy: 0.5393
Epoch 11/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.8595 - accuracy: 0.5373
Epoch 12/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.8564 - accuracy: 0.5363
Epoch 13/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.8533 - accuracy: 0.5343
Epoch 14/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.8502 - accuracy: 0.5353
Epoch 15/100
1005/1005 [==============================] - 0s 64us/sample - loss: 0.8474 - accuracy: 0.5353
Epoch 16/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.8446 - accuracy: 0.5333
Epoch 17/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.8419 - accuracy: 0.5353
Epoch 18/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.8393 - accuracy: 0.5333
Epoch 19/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.8369 - accuracy: 0.5313
Epoch 20/100
1005/1005 [==============================] - 0s 64us/sample - loss: 0.8342 - accuracy: 0.5323
Epoch 21/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.8319 - accuracy: 0.5303
```

```
Epoch 22/100
1005/1005 [==============================] - 0s 76us/sample - loss: 0.8295 - accuracy: 0.5313
Epoch 23/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.8273 - accuracy: 0.5294
Epoch 24/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.8250 - accuracy: 0.5313
Epoch 25/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.8228 - accuracy: 0.5313
Epoch 26/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.8207 - accuracy: 0.5313
Epoch 27/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.8187 - accuracy: 0.5294
Epoch 28/100
1005/1005 [==============================] - 0s 65us/sample - loss: 0.8167 - accuracy: 0.5323
Epoch 29/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.8148 - accuracy: 0.5294
Epoch 30/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.8130 - accuracy: 0.5284
Epoch 31/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.8111 - accuracy: 0.5294
Epoch 32/100
1005/1005 [==============================] - 0s 65us/sample - loss: 0.8094 - accuracy: 0.5294
Epoch 33/100
1005/1005 [==============================] - 0s 65us/sample - loss: 0.8076 - accuracy: 0.5303
Epoch 34/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.8059 - accuracy: 0.5303
Epoch 35/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.8043 - accuracy: 0.5284
Epoch 36/100
1005/1005 [==============================] - 0s 75us/sample - loss: 0.8027 - accuracy: 0.5294
Epoch 37/100
1005/1005 [==============================] - 0s 62us/sample - loss: 0.8011 - accuracy: 0.5274
Epoch 38/100
1005/1005 [==============================] - 0s 60us/sample - loss: 0.7996 - accuracy: 0.5274
Epoch 39/100
1005/1005 [==============================] - 0s 64us/sample - loss: 0.7982 - accuracy: 0.5264
Epoch 40/100
1005/1005 [==============================] - 0s 63us/sample - loss: 0.7968 - accuracy: 0.5264
Epoch 41/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.7954 - accuracy: 0.5264
Epoch 42/100
1005/1005 [==============================] - 0s 60us/sample - loss: 0.7941 - accuracy: 0.5284
Epoch 43/100
```

```
1005/1005 [==============================] - 0s 62us/sample - loss: 0.7927 - accuracy: 0.5264
Epoch 44/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.7915 - accuracy: 0.5264
Epoch 45/100
1005/1005 [==============================] - 0s 60us/sample - loss: 0.7903 - accuracy: 0.5284
Epoch 46/100
1005/1005 [==============================] - 0s 62us/sample - loss: 0.7891 - accuracy: 0.5284
Epoch 47/100
1005/1005 [==============================] - 0s 62us/sample - loss: 0.7879 - accuracy: 0.5274
Epoch 48/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.7868 - accuracy: 0.5284
Epoch 49/100
1005/1005 [==============================] - 0s 64us/sample - loss: 0.7857 - accuracy: 0.5274
Epoch 50/100
1005/1005 [==============================] - 0s 61us/sample - loss: 0.7846 - accuracy: 0.5244
Epoch 51/100
1005/1005 [==============================] - 0s 71us/sample - loss: 0.7835 - accuracy: 0.5254
Epoch 52/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.7826 - accuracy: 0.5254
Epoch 53/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.7816 - accuracy: 0.5244
Epoch 54/100
1005/1005 [==============================] - 0s 62us/sample - loss: 0.7806 - accuracy: 0.5244
Epoch 55/100
1005/1005 [==============================] - 0s 63us/sample - loss: 0.7797 - accuracy: 0.5244
Epoch 56/100
1005/1005 [==============================] - 0s 60us/sample - loss: 0.7788 - accuracy: 0.5264
Epoch 57/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.7779 - accuracy: 0.5234
Epoch 58/100
1005/1005 [==============================] - 0s 63us/sample - loss: 0.7771 - accuracy: 0.5244
Epoch 59/100
1005/1005 [==============================] - 0s 65us/sample - loss: 0.7763 - accuracy: 0.5244
Epoch 60/100
1005/1005 [==============================] - 0s 62us/sample - loss: 0.7754 - accuracy: 0.5224
Epoch 61/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.7747 - accuracy: 0.5214
Epoch 62/100
1005/1005 [==============================] - 0s 71us/sample - loss: 0.7739 - accuracy: 0.5214
Epoch 63/100
1005/1005 [==============================] - 0s 62us/sample - loss: 0.7732 - accuracy: 0.5224
Epoch 64/100
1005/1005 [==============================] - 0s 60us/sample - loss: 0.7725 - accuracy: 0.5224
```

```
Epoch 65/100
1005/1005 [==============================] - 0s 63us/sample - loss: 0.7718 - accuracy: 0.5224
Epoch 66/100
1005/1005 [==============================] - 0s 84us/sample - loss: 0.7711 - accuracy: 0.5214
Epoch 67/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.7704 - accuracy: 0.5214
Epoch 68/100
1005/1005 [==============================] - 0s 73us/sample - loss: 0.7698 - accuracy: 0.5214
Epoch 69/100
1005/1005 [==============================] - 0s 64us/sample - loss: 0.7691 - accuracy: 0.5204
Epoch 70/100
1005/1005 [==============================] - 0s 64us/sample - loss: 0.7685 - accuracy: 0.5204
Epoch 71/100
1005/1005 [==============================] - 0s 61us/sample - loss: 0.7680 - accuracy: 0.5204
Epoch 72/100
1005/1005 [==============================] - 0s 62us/sample - loss: 0.7674 - accuracy: 0.5204
Epoch 73/100
1005/1005 [==============================] - 0s 62us/sample - loss: 0.7668 - accuracy: 0.5204
Epoch 74/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.7662 - accuracy: 0.5204
Epoch 75/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.7657 - accuracy: 0.5194
Epoch 76/100
1005/1005 [==============================] - 0s 60us/sample - loss: 0.7652 - accuracy: 0.5184
Epoch 77/100
1005/1005 [==============================] - 0s 62us/sample - loss: 0.7647 - accuracy: 0.5194
Epoch 78/100
1005/1005 [==============================] - 0s 59us/sample - loss: 0.7642 - accuracy: 0.5204
Epoch 79/100
1005/1005 [==============================] - 0s 61us/sample - loss: 0.7638 - accuracy: 0.5214
Epoch 80/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.7633 - accuracy: 0.5204
Epoch 81/100
1005/1005 [==============================] - 0s 74us/sample - loss: 0.7628 - accuracy: 0.5184
Epoch 82/100
1005/1005 [==============================] - 0s 62us/sample - loss: 0.7624 - accuracy: 0.5194
Epoch 83/100
1005/1005 [==============================] - 0s 59us/sample - loss: 0.7619 - accuracy: 0.5194
Epoch 84/100
1005/1005 [==============================] - 0s 65us/sample - loss: 0.7615 - accuracy: 0.5184
Epoch 85/100
1005/1005 [==============================] - 0s 71us/sample - loss: 0.7612 - accuracy: 0.5204
Epoch 86/100
```

```
1005/1005 [==============================] - 0s 73us/sample - loss: 0.7607 - accuracy: 0.5184
Epoch 87/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.7603 - accuracy: 0.5204
Epoch 88/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.7599 - accuracy: 0.5194
Epoch 89/100
1005/1005 [==============================] - 0s 61us/sample - loss: 0.7596 - accuracy: 0.5174
Epoch 90/100
1005/1005 [==============================] - 0s 60us/sample - loss: 0.7594 - accuracy: 0.5214
Epoch 91/100
1005/1005 [==============================] - 0s 63us/sample - loss: 0.7590 - accuracy: 0.5194
Epoch 92/100
1005/1005 [==============================] - 0s 61us/sample - loss: 0.7587 - accuracy: 0.5184
Epoch 93/100
1005/1005 [==============================] - 0s 63us/sample - loss: 0.7583 - accuracy: 0.5164
Epoch 94/100
1005/1005 [==============================] - 0s 63us/sample - loss: 0.7580 - accuracy: 0.5184
Epoch 95/100
1005/1005 [==============================] - 0s 65us/sample - loss: 0.7577 - accuracy: 0.5164
Epoch 96/100
1005/1005 [==============================] - 0s 74us/sample - loss: 0.7574 - accuracy: 0.5174
Epoch 97/100
1005/1005 [==============================] - 0s 62us/sample - loss: 0.7572 - accuracy: 0.5164
Epoch 98/100
1005/1005 [==============================] - 0s 63us/sample - loss: 0.7569 - accuracy: 0.5164
Epoch 99/100
1005/1005 [==============================] - 0s 61us/sample - loss: 0.7566 - accuracy: 0.5174
Epoch 100/100
1005/1005 [==============================] - 0s 63us/sample - loss: 0.7564 - accuracy: 0.5154

history_list ['loss', 'accuracy']
```

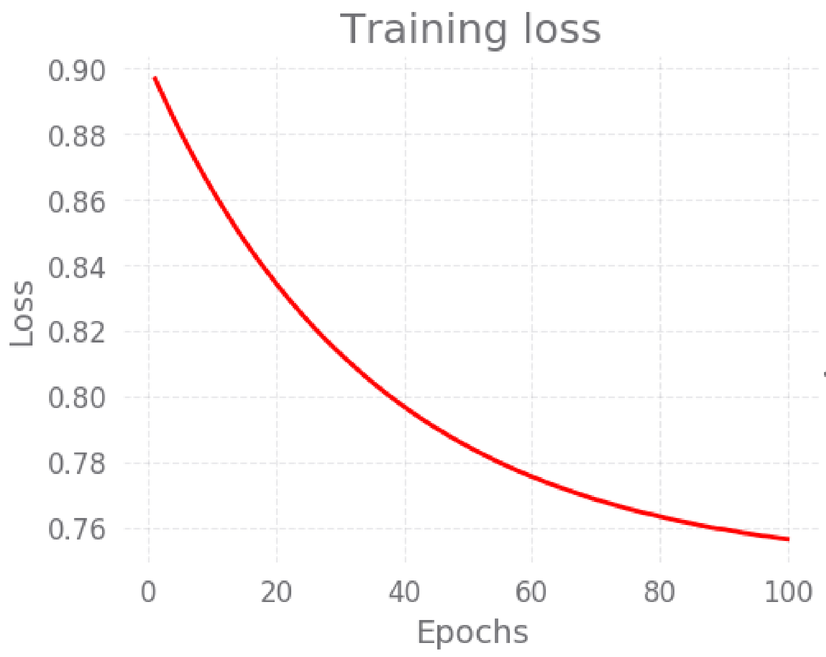
In [0]:

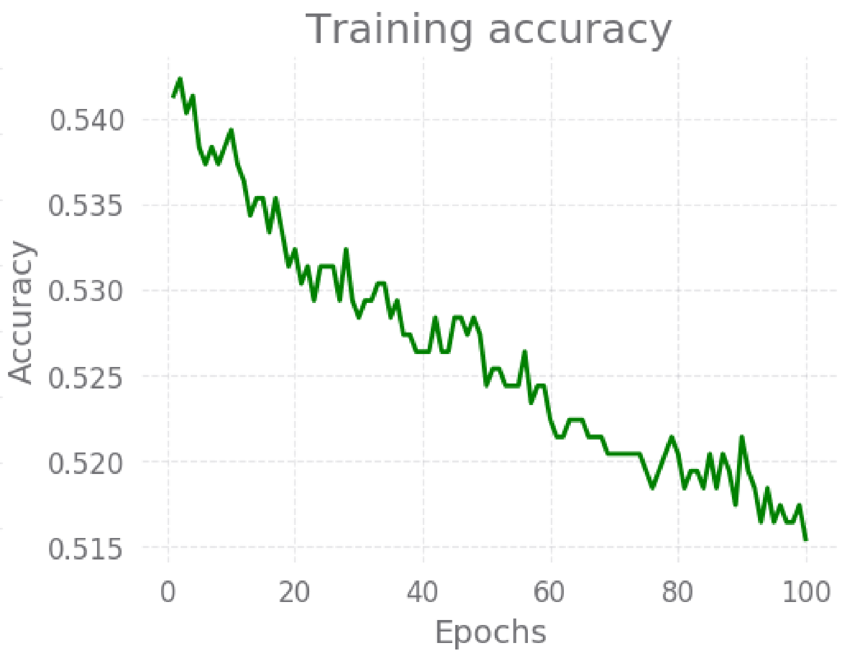
```
#@title ## Evaluating

evaluation = linear_model.evaluate(x_test, y_test_c, verbose=0)

print('Loss: %.3f\t Accuracy: %.2f%%' % (evaluation[0], evaluation[1] * 100))
```

```
Loss: 0.743      Accuracy: 54.55%
```

In [0]:

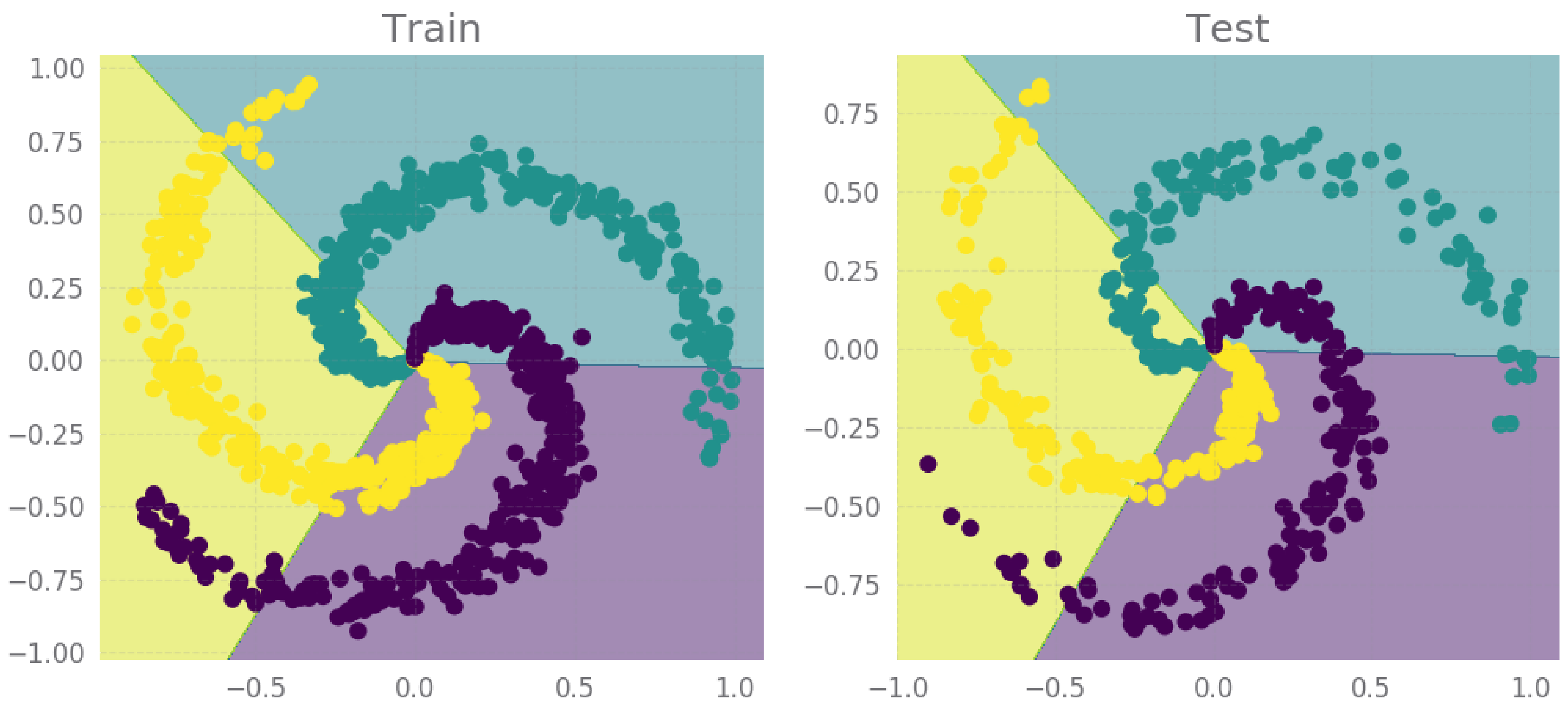
```
#@title ## Plotting

xfig = 12.0 #@param {type:"number"}
yfig = 4.0 #@param {type:"number"}

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(xfig, yfig))


loss = history_dict[history_list[history_list.index('loss')]]
acc = history_dict[history_list[history_list.index('accuracy')]]

epochs = range(1, len(loss) + 1)


ax1.plot(epochs, loss, 'r', label='Training loss')
ax1.set_title('Training loss')
ax1.set_xlabel('Epochs')
ax1.set_ylabel('Loss')

ax2.plot(epochs, acc, 'g', label='Training accuracy')
ax2.set_title('Training accuracy')
ax2.set_xlabel('Epochs')
ax2.set_ylabel('Accuracy')

plt.show()
```


Training loss
Loss
Epochs
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.76
0
20
40
60
80
100


Training accuracy
Accuracy
Epochs
0.540
0.535
0.530
0.525
0.520
0.515
0
20
40
60
80
100

In [0]:
```
#@title ## Multiclass decision boundary

def plot_mc_decision_boundary(model, x, y, steps=1000):
    """ Plot multiclass decision boundary,

    Args:
        model (keras.Model): a keras model.
        x (ndarray): a numpy array.
        y (ndarray): a numpy array.
        steps (integer): number of linear spaces.

    Returns:
        None
    """
    x_min, x_max = x[:, 0].min() - 0.1, x[:, 0].max() + 0.1
    y_min, y_max = x[:, 1].min() - 0.1, x[:, 1].max() + 0.1

    x_span = np.linspace(x_min, x_max, steps)
    y_span = np.linspace(y_min, y_max, steps)
    xx, yy = np.meshgrid(x_span, y_span)

    y_pred = model.predict(np.c_[xx.ravel(), yy.ravel()])
    y_pred = np.argmax(y_pred, axis=1)
    y_pred = y_pred.reshape(xx.shape)

    plt.contourf(xx, yy, y_pred, alpha=0.5)
    plt.scatter(x[:, 0], x[:, 1], c=y)
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())

    return None
```

In [0]:
```
#@title ## Visualize the decision boundary

xfig = 12.0 #@param {type:"number"}
yfig = 5.0 #@param {type:"number"}

plt.figure(figsize=(xfig, yfig))
plt.subplot(1, 2, 1)
plt.title('Train')
plot_mc_decision_boundary(model=linear_model, x=x_train, y=y_train)
plt.subplot(1, 2, 2)
plt.title('Test')
plot_mc_decision_boundary(model=linear_model, x=x_test, y=y_test)
plt.show()
```



In [0]:
```
#@title ## Plot confusion matrix

def plot_confusion_matrix(cm, classes):
    """ Plot confusion matrix

    Args:
        cm (ndarray): the input of confusion matrix.
        classes (integer): number of classes.

    Returns:
        None
    """
    cmap=plt.cm.Blues
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title("Confusion Matrix")
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    plt.grid(False)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt), horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()

    return None
```

Classification Report

`sklearn.metrics.classification_report`

Build a text report showing the main classification metrics

Params:

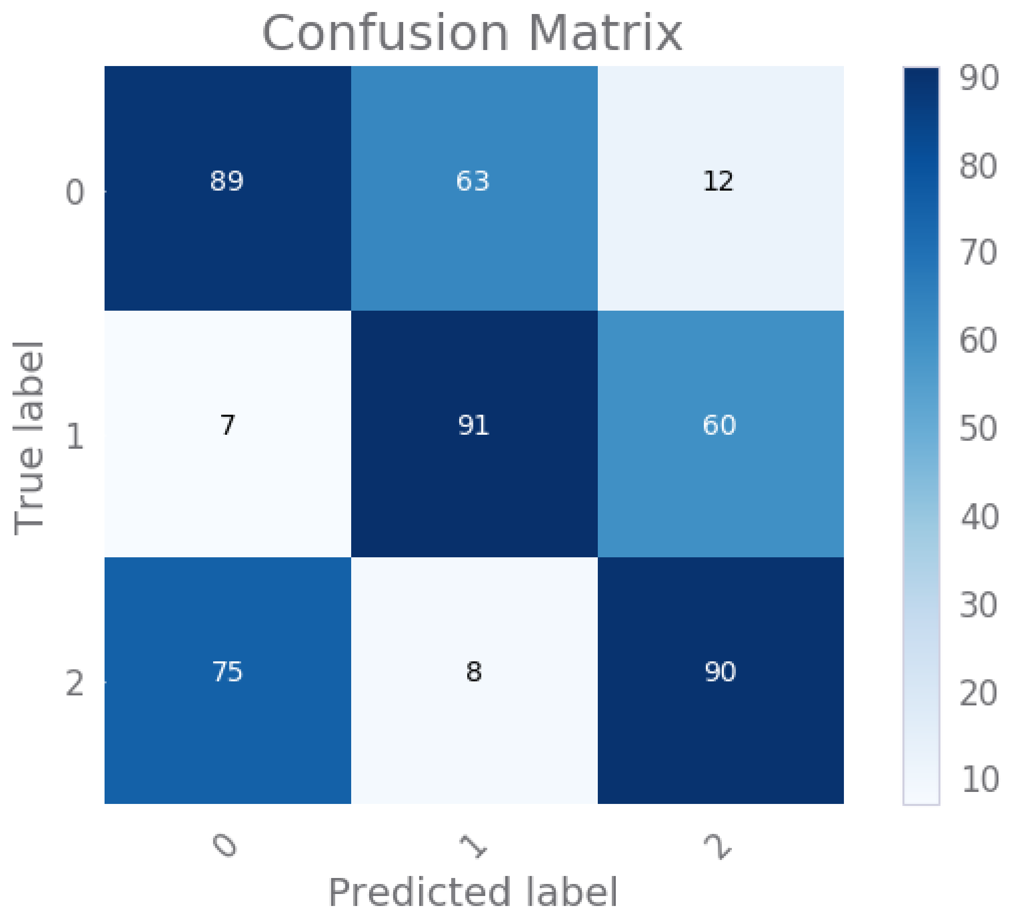
- `y_true` : 1d array-like, or label indicator array / sparse matrix
- `y_pred` : 1d array-like, or label indicator array / sparse matrix

Output:

```
precision    recall  f1-score   support

          1       1.00      0.67      0.80         3
          2       0.00      0.00      0.00         0
          3       0.00      0.00      0.00         0

  micro avg       1.00      0.67      0.80         3
  macro avg       0.33      0.22      0.27         3
weighted avg       1.00      0.67      0.80         3
```

In [0]:
```
#@title ## Confusion matrix

pred_test = linear_model.predict(x_test)
pred_test = np.argmax(pred_test, axis=1)

cm = confusion_matrix(y_test, pred_test)
plot_confusion_matrix(cm=cm, classes=[0, 1, 2])

print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

           0       0.52      0.54      0.53       164
           1       0.56      0.58      0.57       158
           2       0.56      0.52      0.54       173

    accuracy                           0.55       495
   macro avg       0.55      0.55      0.55       495
weighted avg       0.55      0.55      0.55       495
```



Non-linear Model

Now let's see how the MLP performs on the data. Note that the only difference is the addition of the non-linear activation function (we use $ReLU$ which is just $max(0, z)$).

In [0]:
```
#@title ## Build non-linear model [MLP Model](https://en.wikipedia.org/wiki/Multilayer_perceptron)

def build_non_linear(n_classes, n_units, n_features, compiling=True, opt='adam', loss='categorical_cr
ossentropy', metrics=None):
    """ Build a linear model [MLP Model].

    Args:
        n_classes (int): dimensionality of the output space.
        n_units (int): dimensionality of the hidden layer.
        n_features (int): indicates that the expected input will be batches of n_features-dimensional
vectors.
        compiling (boolean): whether the model complied or not.
        opt (str, keras.optimizers): optimizer instance.
        loss (str, keras.losses): objective function.
        metrics (str, keras.metrics): List of metrics to be evaluated by the model during training an
d testing.

    Returns:
        model (keras.Model): a keras model.
    """

    model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(n_features,)),
        tf.keras.layers.Dense(n_units, activation='relu'),
        tf.keras.layers.Dense(n_classes, activation='softmax')
    ])

    if compiling:

        if metrics is None or not isinstance(metrics, list):
            metrics = ['accuracy']

        model.compile(optimizer=opt, loss=loss, metrics=metrics)

    return model
```

In [0]:
```
#@title ## Create non-linear model

compiling = True #@param {type:"boolean"}

n_units = 5 #@param {type:"integer"}
opt = 'adam' #@param {type:"string"}
loss = 'categorical_crossentropy' #@param {type:"string"}
metrics = "accuracy" #@param ["accuracy"] {allow-input: true}

n_classes = y_train_c.shape[1] # 3, num_classes
n_features = x_train.shape[1] # 2, dimensions


non_linear_model = build_non_linear(n_classes, n_units, n_features, compiling, opt, loss, metrics)
non_linear_model
```

Out[0]:
```
<tensorflow.python.keras.engine.sequential.Sequential at 0x7fa460436d68>
```

In [0]:
```
#@title ## Fit non-linear model.

r = non_linear_model.fit(x_train, y_train_c, epochs=100, verbose=1)

history_dict = r.history
history_list = list(history_dict.keys())

print()
print('history_list', history_list)
```

```
Train on 1005 samples
Epoch 1/100
1005/1005 [==============================] - 0s 165us/sample - loss: 1.1430 - accuracy: 0.1254
Epoch 2/100
1005/1005 [==============================] - 0s 72us/sample - loss: 1.1159 - accuracy: 0.1701
Epoch 3/100
1005/1005 [==============================] - 0s 72us/sample - loss: 1.0931 - accuracy: 0.2259
Epoch 4/100
1005/1005 [==============================] - 0s 67us/sample - loss: 1.0735 - accuracy: 0.2876
Epoch 5/100
1005/1005 [==============================] - 0s 66us/sample - loss: 1.0560 - accuracy: 0.3493
Epoch 6/100
1005/1005 [==============================] - 0s 68us/sample - loss: 1.0410 - accuracy: 0.4030
Epoch 7/100
1005/1005 [==============================] - 0s 68us/sample - loss: 1.0285 - accuracy: 0.4488
Epoch 8/100
1005/1005 [==============================] - 0s 69us/sample - loss: 1.0179 - accuracy: 0.5264
Epoch 9/100
1005/1005 [==============================] - 0s 66us/sample - loss: 1.0084 - accuracy: 0.5403
Epoch 10/100
1005/1005 [==============================] - 0s 65us/sample - loss: 0.9998 - accuracy: 0.5284
Epoch 11/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.9917 - accuracy: 0.5214
Epoch 12/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.9838 - accuracy: 0.5264
Epoch 13/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.9758 - accuracy: 0.5313
Epoch 14/100
1005/1005 [==============================] - 0s 74us/sample - loss: 0.9677 - accuracy: 0.5373
Epoch 15/100
1005/1005 [==============================] - 0s 75us/sample - loss: 0.9591 - accuracy: 0.5333
Epoch 16/100
1005/1005 [==============================] - 0s 74us/sample - loss: 0.9503 - accuracy: 0.5294
Epoch 17/100
1005/1005 [==============================] - 0s 81us/sample - loss: 0.9412 - accuracy: 0.5333
Epoch 18/100
1005/1005 [==============================] - 0s 76us/sample - loss: 0.9312 - accuracy: 0.5333
Epoch 19/100
1005/1005 [==============================] - 0s 73us/sample - loss: 0.9215 - accuracy: 0.5333
Epoch 20/100
1005/1005 [==============================] - 0s 75us/sample - loss: 0.9111 - accuracy: 0.5353
Epoch 21/100
1005/1005 [==============================] - 0s 75us/sample - loss: 0.9005 - accuracy: 0.5353
```

```
Epoch 22/100
1005/1005 [==============================] - 0s 75us/sample - loss: 0.8897 - accuracy: 0.5333
Epoch 23/100
1005/1005 [==============================] - 0s 75us/sample - loss: 0.8786 - accuracy: 0.5224
Epoch 24/100
1005/1005 [==============================] - 0s 78us/sample - loss: 0.8675 - accuracy: 0.5234
Epoch 25/100
1005/1005 [==============================] - 0s 76us/sample - loss: 0.8567 - accuracy: 0.5294
Epoch 26/100
1005/1005 [==============================] - 0s 75us/sample - loss: 0.8457 - accuracy: 0.5234
Epoch 27/100
1005/1005 [==============================] - 0s 73us/sample - loss: 0.8350 - accuracy: 0.5284
Epoch 28/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.8245 - accuracy: 0.5244
Epoch 29/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.8144 - accuracy: 0.5284
Epoch 30/100
1005/1005 [==============================] - 0s 84us/sample - loss: 0.8047 - accuracy: 0.5244
Epoch 31/100
1005/1005 [==============================] - 0s 71us/sample - loss: 0.7951 - accuracy: 0.5234
Epoch 32/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.7862 - accuracy: 0.5254
Epoch 33/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.7775 - accuracy: 0.5323
Epoch 34/100
1005/1005 [==============================] - 0s 75us/sample - loss: 0.7694 - accuracy: 0.5363
Epoch 35/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.7617 - accuracy: 0.5363
Epoch 36/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.7544 - accuracy: 0.5403
Epoch 37/100
1005/1005 [==============================] - 0s 76us/sample - loss: 0.7476 - accuracy: 0.5413
Epoch 38/100
1005/1005 [==============================] - 0s 76us/sample - loss: 0.7410 - accuracy: 0.5453
Epoch 39/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.7345 - accuracy: 0.5483
Epoch 40/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.7285 - accuracy: 0.5483
Epoch 41/100
1005/1005 [==============================] - 0s 72us/sample - loss: 0.7225 - accuracy: 0.5542
Epoch 42/100
1005/1005 [==============================] - 0s 74us/sample - loss: 0.7170 - accuracy: 0.5562
Epoch 43/100
```

```
1005/1005 [==============================] - 0s 81us/sample - loss: 0.7121 - accuracy: 0.5572
Epoch 44/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.7070 - accuracy: 0.5662
Epoch 45/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.7021 - accuracy: 0.5632
Epoch 46/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.6977 - accuracy: 0.5711
Epoch 47/100
1005/1005 [==============================] - 0s 72us/sample - loss: 0.6935 - accuracy: 0.5791
Epoch 48/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.6893 - accuracy: 0.5781
Epoch 49/100
1005/1005 [==============================] - 0s 65us/sample - loss: 0.6852 - accuracy: 0.5811
Epoch 50/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.6815 - accuracy: 0.5791
Epoch 51/100
1005/1005 [==============================] - 0s 71us/sample - loss: 0.6779 - accuracy: 0.5851
Epoch 52/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.6744 - accuracy: 0.5851
Epoch 53/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.6712 - accuracy: 0.5841
Epoch 54/100
1005/1005 [==============================] - 0s 71us/sample - loss: 0.6683 - accuracy: 0.5831
Epoch 55/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.6648 - accuracy: 0.5910
Epoch 56/100
1005/1005 [==============================] - 0s 74us/sample - loss: 0.6622 - accuracy: 0.5920
Epoch 57/100
1005/1005 [==============================] - 0s 77us/sample - loss: 0.6593 - accuracy: 0.5930
Epoch 58/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.6564 - accuracy: 0.5950
Epoch 59/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.6534 - accuracy: 0.5960
Epoch 60/100
1005/1005 [==============================] - 0s 77us/sample - loss: 0.6509 - accuracy: 0.5970
Epoch 61/100
1005/1005 [==============================] - 0s 73us/sample - loss: 0.6482 - accuracy: 0.6030
Epoch 62/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.6457 - accuracy: 0.6060
Epoch 63/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.6429 - accuracy: 0.6100
Epoch 64/100
1005/1005 [==============================] - 0s 78us/sample - loss: 0.6406 - accuracy: 0.6100
```

```
Epoch 65/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.6380 - accuracy: 0.6139
Epoch 66/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.6359 - accuracy: 0.6139
Epoch 67/100
1005/1005 [==============================] - 0s 71us/sample - loss: 0.6333 - accuracy: 0.6169
Epoch 68/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.6309 - accuracy: 0.6169
Epoch 69/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.6287 - accuracy: 0.6229
Epoch 70/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.6264 - accuracy: 0.6259
Epoch 71/100
1005/1005 [==============================] - 0s 78us/sample - loss: 0.6242 - accuracy: 0.6299
Epoch 72/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.6219 - accuracy: 0.6388
Epoch 73/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.6198 - accuracy: 0.6378
Epoch 74/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.6176 - accuracy: 0.6408
Epoch 75/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.6157 - accuracy: 0.6468
Epoch 76/100
1005/1005 [==============================] - 0s 65us/sample - loss: 0.6130 - accuracy: 0.6507
Epoch 77/100
1005/1005 [==============================] - 0s 73us/sample - loss: 0.6113 - accuracy: 0.6468
Epoch 78/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.6088 - accuracy: 0.6567
Epoch 79/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.6068 - accuracy: 0.6587
Epoch 80/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.6047 - accuracy: 0.6537
Epoch 81/100
1005/1005 [==============================] - 0s 76us/sample - loss: 0.6026 - accuracy: 0.6627
Epoch 82/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.6007 - accuracy: 0.6647
Epoch 83/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.5988 - accuracy: 0.6736
Epoch 84/100
1005/1005 [==============================] - 0s 65us/sample - loss: 0.5966 - accuracy: 0.6746
Epoch 85/100
1005/1005 [==============================] - 0s 75us/sample - loss: 0.5944 - accuracy: 0.6746
Epoch 86/100
```

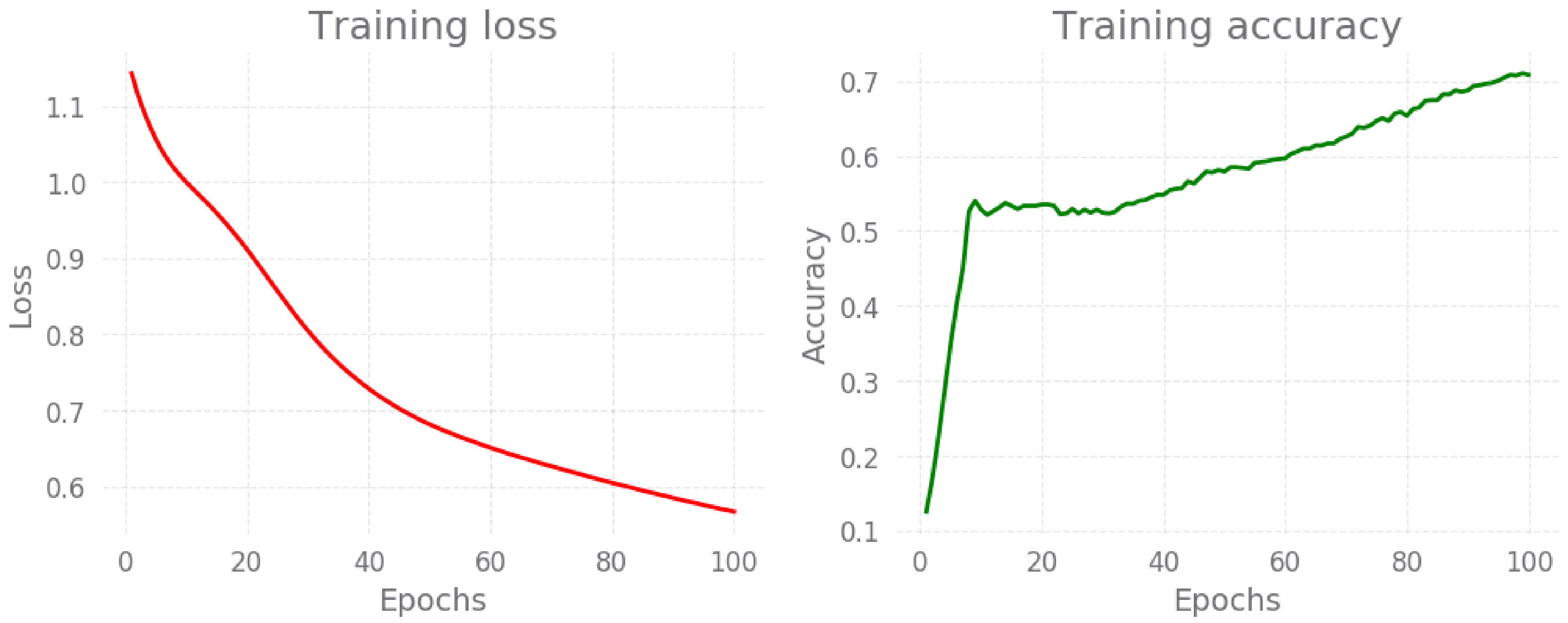
```
1005/1005 [==============================] - 0s 69us/sample - loss: 0.5925 - accuracy: 0.6826
Epoch 87/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.5907 - accuracy: 0.6826
Epoch 88/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.5885 - accuracy: 0.6876
Epoch 89/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.5870 - accuracy: 0.6856
Epoch 90/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.5847 - accuracy: 0.6876
Epoch 91/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.5828 - accuracy: 0.6935
Epoch 92/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.5810 - accuracy: 0.6945
Epoch 93/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.5792 - accuracy: 0.6965
Epoch 94/100
1005/1005 [==============================] - 0s 66us/sample - loss: 0.5777 - accuracy: 0.6975
Epoch 95/100
1005/1005 [==============================] - 0s 69us/sample - loss: 0.5755 - accuracy: 0.7005
Epoch 96/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.5737 - accuracy: 0.7045
Epoch 97/100
1005/1005 [==============================] - 0s 70us/sample - loss: 0.5721 - accuracy: 0.7085
Epoch 98/100
1005/1005 [==============================] - 0s 67us/sample - loss: 0.5701 - accuracy: 0.7075
Epoch 99/100
1005/1005 [==============================] - 0s 80us/sample - loss: 0.5687 - accuracy: 0.7104
Epoch 100/100
1005/1005 [==============================] - 0s 68us/sample - loss: 0.5668 - accuracy: 0.7085

history_list ['loss', 'accuracy']
```

In [0]:

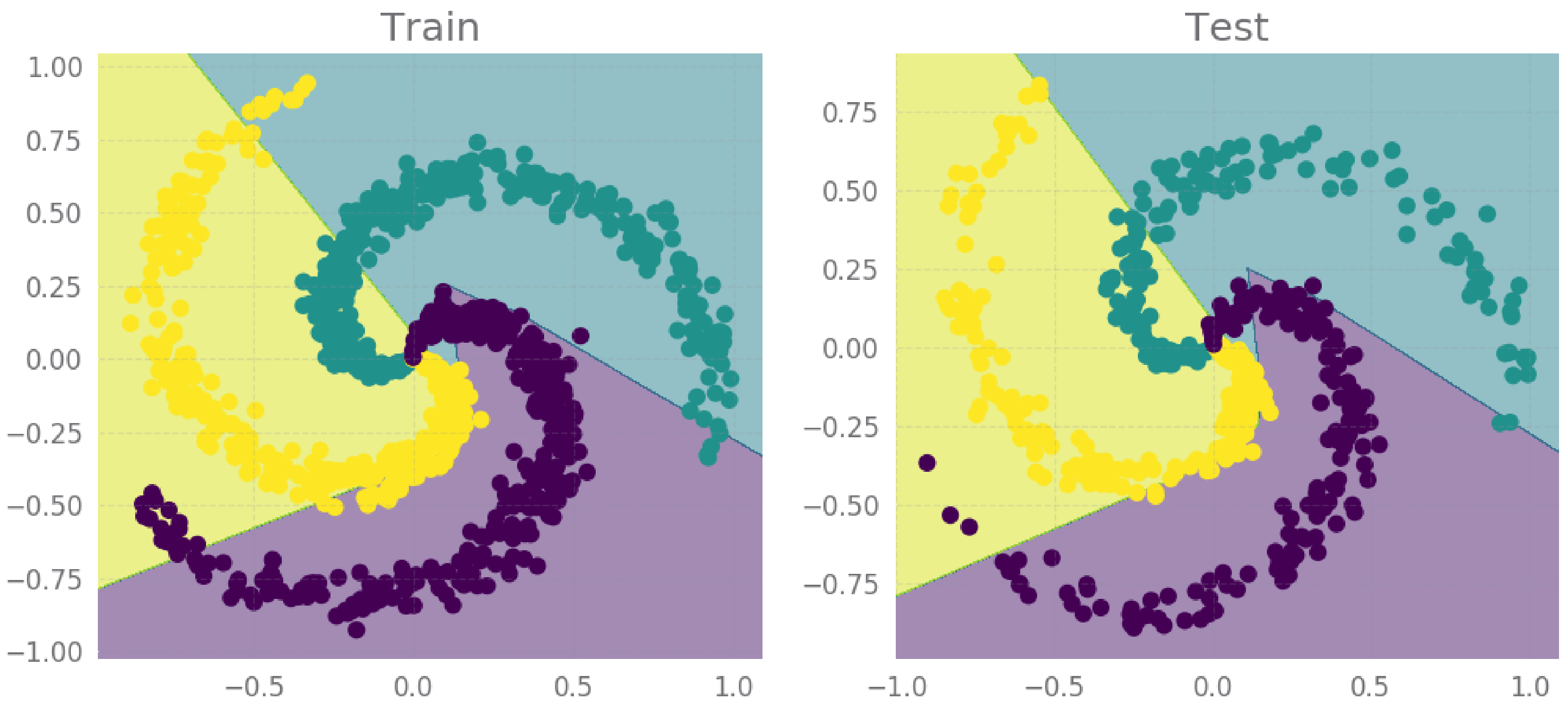
```python
#@title ## Evaluating

evaluation = non_linear_model.evaluate(x_test, y_test_c, verbose=0)

print('Loss: %.3f\t Accuracy: %.2f%%' % (evaluation[0], evaluation[1] * 100))
```

```
Loss: 0.558      Accuracy: 73.13%
```

In [0]:
```
#@title ## Plotting

xfig = 12.0 #@param {type:"number"}
yfig = 4.0 #@param {type:"number"}

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(xfig, yfig))


loss = history_dict[history_list[history_list.index('loss')]]
acc = history_dict[history_list[history_list.index('accuracy')]]

epochs = range(1, len(loss) + 1)


ax1.plot(epochs, loss, 'r', label='Training loss')
ax1.set_title('Training loss')
ax1.set_xlabel('Epochs')
ax1.set_ylabel('Loss')

ax2.plot(epochs, acc, 'g', label='Training accuracy')
ax2.set_title('Training accuracy')
ax2.set_xlabel('Epochs')
ax2.set_ylabel('Accuracy')

plt.show()
```

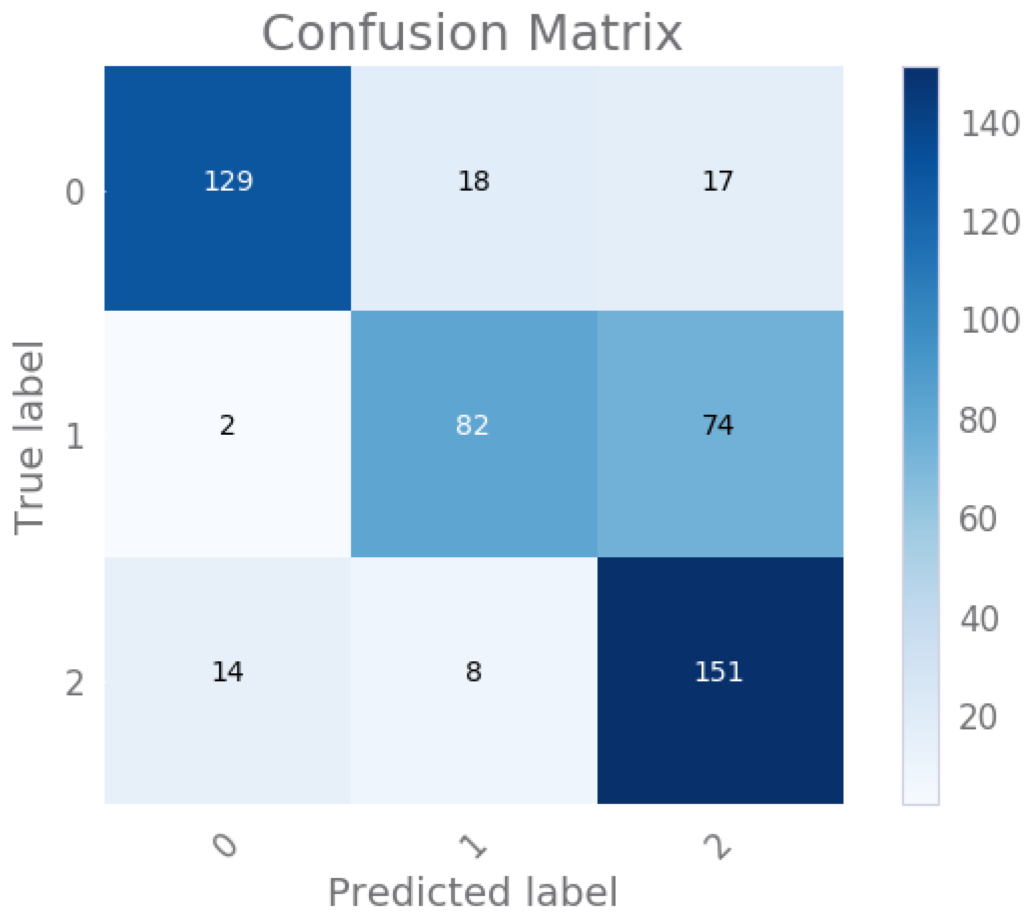

Training loss
Loss
1.1
1.0
0.9
0.8
0.7
0.6
0
20
40
60
80
100
Epochs
Training accuracy
Accuracy
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100
Epochs

In [0]:
```
#@title ## Visualize the decision boundary

xfig = 12.0 #@param {type:"number"}
yfig = 5.0 #@param {type:"number"}

plt.figure(figsize=(xfig, yfig))
plt.subplot(1, 2, 1)
plt.title('Train')
plot_mc_decision_boundary(model=non_linear_model, x=x_train, y=y_train)
plt.subplot(1, 2, 2)
plt.title('Test')
plot_mc_decision_boundary(model=non_linear_model, x=x_test, y=y_test)
plt.show()
```



In [0]:
```
#@title ## Confusion matrix

pred_test = non_linear_model.predict(x_test)
pred_test = np.argmax(pred_test, axis=1)

cm = confusion_matrix(y_test, pred_test)
plot_confusion_matrix(cm=cm, classes=[0, 1, 2])

print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

           0       0.89      0.79      0.83       164
           1       0.76      0.52      0.62       158
           2       0.62      0.87      0.73       173

    accuracy                           0.73       495
   macro avg       0.76      0.73      0.73       495
weighted avg       0.76      0.73      0.73       495
```



Overfitting

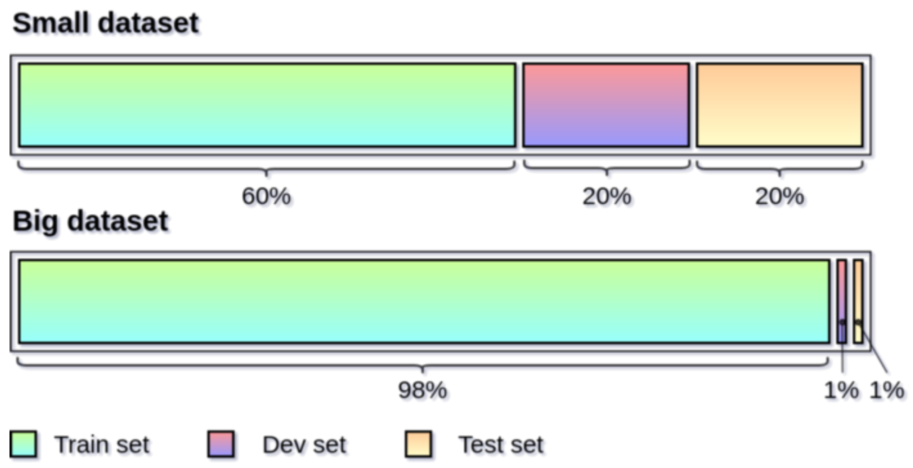
Though neural networks are great at capturing non-linear relationships they are highly susceptible to overfitting to the training data and failing to generalize on test data. Just take a look at the example below where we generate completely random data and are able to fit a model with $2 * N * C + D$ (https://arxiv.org/abs/1611.03530) hidden units. The training performance is great but the overfitting leads to very poor test performance. We'll be covering strategies to tackle overfitting in future lessons.

Let me define overfitting more formally. Overfitting refers to a model that models the “training data” too well. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.

How do you know your NN is overfitting?

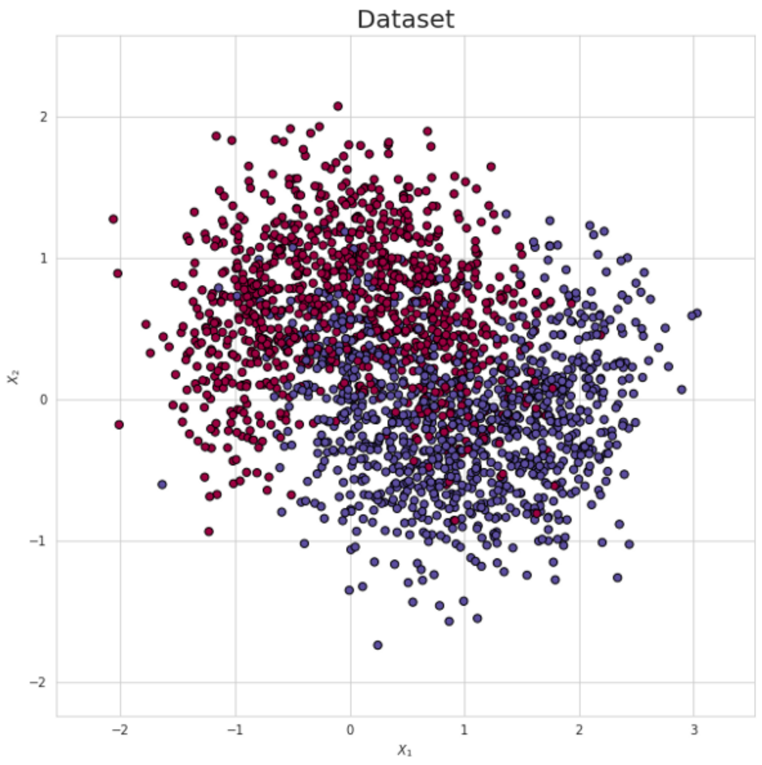
In practice, detecting that our model is overfitting is difficult. It’s not uncommon that our trained model is already in production and then we start to realize that something is wrong. In fact, it is only by confronting new data that you can make sure that everything is working properly. However, during the training we should try to reproduce the real conditions as much as possible. For this reason, it is good practice to divide our dataset into three parts - training set, dev set (also known as cross-validation or hold-out) and test set. Our model learns by seeing only the first of these parts. Hold-out is used to track our progress and draw conclusions to optimise the model. While, we use a test set at the end of the training process to evaluate the performance of our model. Using completely new data allows us to get an unbiased opinion on how well our algorithm works.

It is very important to make sure that your cross-validation and test set come from the same distribution as well as that they accurately reflect data that we expect to receive in the future. Only then we can be sure that the decisions we make during the learning process bring us closer to a better solution. I know what you are thinking about… “How should I divide my dataset?” Until recently, one of the most frequently recommended splits was 60/20/20, but in the era of big data, when our dataset can count millions of entries, those fixed proportions are no longer appropriate. In short, everything depends on the size of the dataset we work with. If we have millions of entries at our disposal, perhaps it would be better idea to divide them in 98/1/1 ratio. Our dev and test sets should be simply large enough to give us high confidence in the performance of our model.

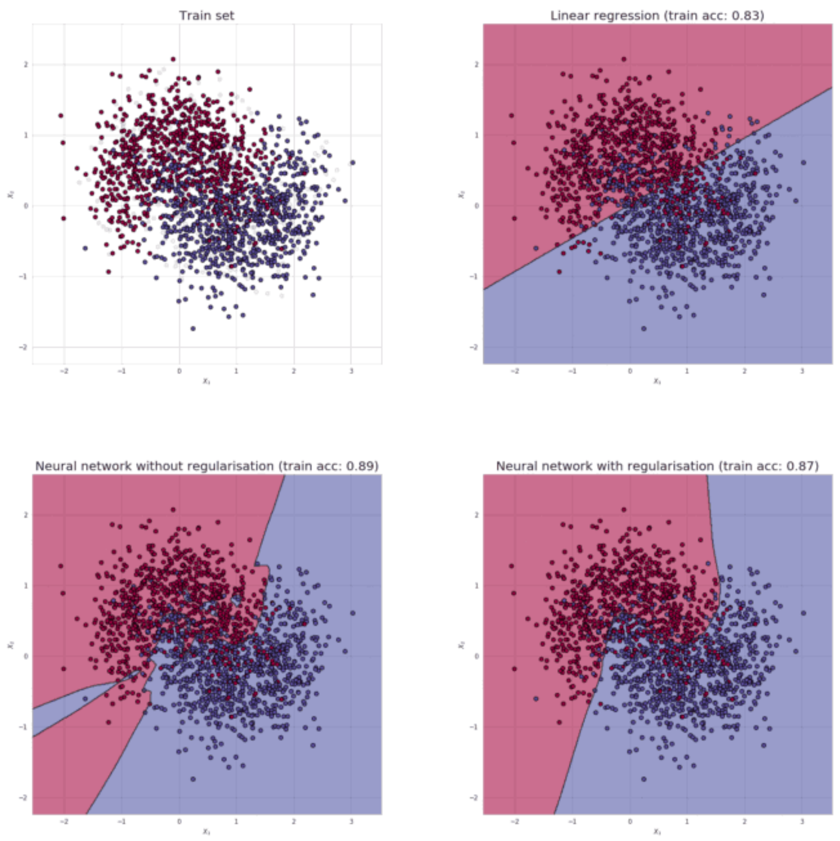


Bias and Variance

To give us a better understanding of this some how complex issue, we will use a simple example, that hopefully allow us to develop a valuable intuition. Our dataset consisting of two classes of points, located in a two-dimensional space.



The first model in the top right corner is very simple and therefore has a high bias, i.e. it is not able to find all significant links between features and result. This is understandable - our dataset has a lot of noise in it, and therefore simple linear regression, is not able to deal with it effectively. Neural networks performed much better, but the first one (shown in the lower left corner) fitted into the data too closely, which made it work significantly worse on the hold-out set. This means that it has a high variance - it fits into the noise and not into the intended output. This undesirable effect was mitigated, in the last model, by the use of regularisation.



Ways to prevent overfitting

L1 and L2 Regularizations

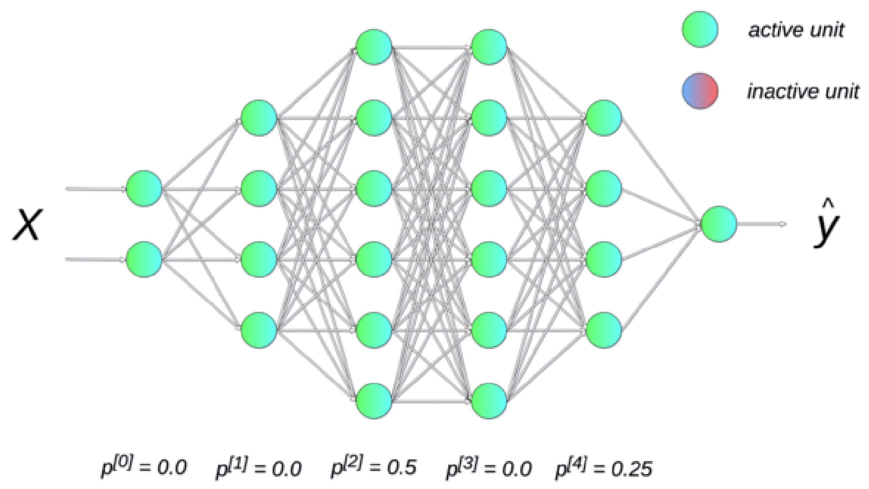
One of the first methods we should try when we need to reduce overfitting is regularisation. It involves adding an extra element to the loss function, which punishes our model for being too complex or, in simple words, for using too high values in the weight matrix. This way we try to limit its flexibility, but also encourage it to build solutions based on multiple features. Two popular versions of this method are L1 - Least Absolute Deviations (LAD) and L2 - Least Square Errors (LS). Equations describing these regularisations are given below. In most cases the use of L1 is preferable, because it reduces the weight values of less important features to zero, very often eliminating them completely from the calculations. In a way, it is a built-in mechanism for automatic featur selection. Moreover, L2 does not perform very well on datasets with a large number of outliers. The use of value squares results in the model minimizing the impact of outliers at the expense of more popular examples.

$$
\begin{aligned}
J_{L1}(W,b) &= \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) + \lambda\|w\|_1 \quad & \|w\|_1 &= \sum_{j=1}^{n_x} |w_j| \\
J_{L2}(W,b) &= \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) + \lambda\|w\|_2 \quad & \|w\|_2 &= \sum_{j=1}^{n_x} w_j^2
\end{aligned}
$$

Increasing the λ value also increases the regularisation effect. Models with a very low λ coefficient value are very "turbulent".

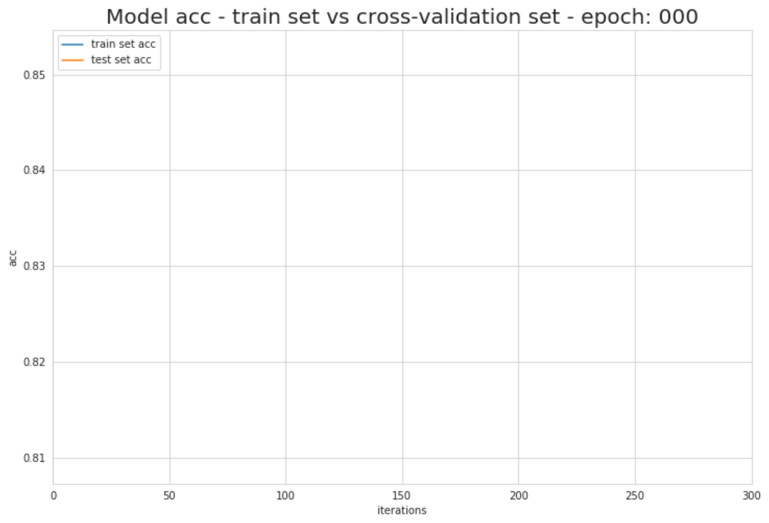
Dropout

Another very popular method of regularization of neural networks is dropout. This idea is actually very simple - every unit of our neural network (except those belonging to the output layer) is given the probability p of being temporarily ignored in calculations. Hyper parameter p is called dropout rate and very often its default value is set to 0.5. Then, in each iteration, we randomly select the neurons that we drop according to the assigned probability. As a result, each time we work with a smaller neural network. The visualization below shows an example of a neural network subjected to a dropout. We can see how in each iteration random neurons from second and fourth layer are deactivated.



Early Stopping

The graph below shows the change in accuracy values calculated on the test and cross-validation sets during subsequent iterations of learning process. We see right away that the model we get at the end is not the best we could have possibly create. To be honest, it is much worse than what we have had after 150 epochs. Why not interrupt the learning process before the model starts overfitting? This observation inspired one of the popular overfitting reduction method, namely early stopping.


Model acc - train set vs cross-validation set - epoch: 000
train set acc
test set acc
acc
iterations

In [0]:
```python
#@title ## Generate random x, y

ov_num_samples = 40 #@param {type:"integer"}
ov_dimensions = 2 #@param {type:"integer"}
ov_num_classes = 3 #@param {type:"integer"}

ov_x = np.random.randn(ov_num_samples * ov_num_classes, ov_dimensions)
ov_y = np.array([[i] * ov_num_samples for i in range(ov_num_classes)])
ov_y = ov_y.flatten()


print()
print('x: %s' % str(ov_x.shape))
print('y: %s' % str(ov_y.shape))
```

```
x: (120, 2)
y: (120,)
```

In [0]:
```
#@title ## Preprocessing

test_size = 0.33 #@param {type:"number"}
shuffling = True #@param {type:"boolean"}

ov_x_train, ov_x_test, ov_y_train, ov_y_test = train_test_split(ov_x, ov_y, test_size=test_size, rand
om_state=RANDOM_SEED, shuffle=shuffling)

print('Train: [x]=> %s,\t [y]=> %s' % (ov_x_train.shape, ov_y_train.shape))
print('Test:  [x]=> %s,\t [y]=> %s' % (ov_x_test.shape, ov_y_test.shape))


ov_y_train_c = tf.keras.utils.to_categorical(ov_y_train, ov_num_classes)
ov_y_test_c = tf.keras.utils.to_categorical(ov_y_test, ov_num_classes)

print('Train: [x]=> %s,\t [y]=> %s' % (ov_x_train.shape, ov_y_train_c.shape))
print('Test:  [x]=> %s,\t [y]=> %s' % (ov_x_test.shape, ov_y_test_c.shape))
```

```
Train: [x]=> (80, 2),    [y]=> (80,)
Test:  [x]=> (40, 2),    [y]=> (40,)
Train: [x]=> (80, 2),    [y]=> (80, 3)
Test:  [x]=> (40, 2),    [y]=> (40, 3)
```

In [0]:
```
#@title ## Create non-linear model

compiling = True #@param {type:"boolean"}

ov_n_units = 2 * num_samples * num_classes + dimensions
opt = 'adam' #@param {type:"string"}
loss = 'categorical_crossentropy' #@param {type:"string"}
metrics = "accuracy" #@param ["accuracy"] {allow-input: true}

ov_n_classes = ov_y_train_c.shape[1] # 3, num_classes
ov_n_features = ov_x_train.shape[1] # 2, dimensions


overfit_model = build_non_linear(ov_n_classes, ov_n_units, ov_n_features, compiling, opt, loss, metri
cs)
overfit_model
```

Out[0]:
```
<tensorflow.python.keras.engine.sequential.Sequential at 0x7fa4532092b0>
```

In [0]:
```
#@title ## Fit overfie model.

r = overfit_model.fit(ov_x_train, ov_y_train_c, epochs=1000, verbose=1)

history_dict = r.history
history_list = list(history_dict.keys())

print()
print('history_list', history_list)
```

```
Train on 80 samples
Epoch 1/1000
80/80 [==============================] - 0s 1ms/sample - loss: 1.1061 - accuracy: 0.2875
Epoch 2/1000
80/80 [==============================] - 0s 124us/sample - loss: 1.0830 - accuracy: 0.5125
Epoch 3/1000
80/80 [==============================] - 0s 112us/sample - loss: 1.0678 - accuracy: 0.5125
Epoch 4/1000
80/80 [==============================] - 0s 116us/sample - loss: 1.0624 - accuracy: 0.5375
Epoch 5/1000
80/80 [==============================] - 0s 116us/sample - loss: 1.0548 - accuracy: 0.5750
Epoch 6/1000
80/80 [==============================] - 0s 114us/sample - loss: 1.0500 - accuracy: 0.5875
Epoch 7/1000
80/80 [==============================] - 0s 110us/sample - loss: 1.0448 - accuracy: 0.5625
Epoch 8/1000
80/80 [==============================] - 0s 116us/sample - loss: 1.0417 - accuracy: 0.5500
Epoch 9/1000
80/80 [==============================] - 0s 130us/sample - loss: 1.0373 - accuracy: 0.5875
Epoch 10/1000
80/80 [==============================] - 0s 110us/sample - loss: 1.0324 - accuracy: 0.5750
Epoch 11/1000
80/80 [==============================] - 0s 118us/sample - loss: 1.0291 - accuracy: 0.5750
Epoch 12/1000
80/80 [==============================] - 0s 116us/sample - loss: 1.0244 - accuracy: 0.5625
Epoch 13/1000
80/80 [==============================] - 0s 109us/sample - loss: 1.0204 - accuracy: 0.5750
Epoch 14/1000
80/80 [==============================] - 0s 125us/sample - loss: 1.0184 - accuracy: 0.5500
Epoch 15/1000
80/80 [==============================] - 0s 110us/sample - loss: 1.0133 - accuracy: 0.5625
Epoch 16/1000
80/80 [==============================] - 0s 110us/sample - loss: 1.0115 - accuracy: 0.6125
Epoch 17/1000
80/80 [==============================] - 0s 112us/sample - loss: 1.0079 - accuracy: 0.6000
Epoch 18/1000
80/80 [==============================] - 0s 122us/sample - loss: 1.0048 - accuracy: 0.6000
Epoch 19/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.9981 - accuracy: 0.6000
Epoch 20/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.9965 - accuracy: 0.5750
Epoch 21/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.9912 - accuracy: 0.6000
```

```
Epoch 22/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.9916 - accuracy: 0.6000
Epoch 23/1000
80/80 [==============================] - 0s 147us/sample - loss: 0.9909 - accuracy: 0.5375
Epoch 24/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.9862 - accuracy: 0.5375
Epoch 25/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.9833 - accuracy: 0.5250
Epoch 26/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.9783 - accuracy: 0.5750
Epoch 27/1000
80/80 [==============================] - 0s 163us/sample - loss: 0.9744 - accuracy: 0.6000
Epoch 28/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.9731 - accuracy: 0.6000
Epoch 29/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.9692 - accuracy: 0.5875
Epoch 30/1000
80/80 [==============================] - 0s 112us/sample - loss: 0.9666 - accuracy: 0.6000
Epoch 31/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.9632 - accuracy: 0.6125
Epoch 32/1000
80/80 [==============================] - 0s 156us/sample - loss: 0.9590 - accuracy: 0.5750
Epoch 33/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.9563 - accuracy: 0.6250
Epoch 34/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.9534 - accuracy: 0.6625
Epoch 35/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.9504 - accuracy: 0.6500
Epoch 36/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.9461 - accuracy: 0.6250
Epoch 37/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.9426 - accuracy: 0.6125
Epoch 38/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.9402 - accuracy: 0.6125
Epoch 39/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.9379 - accuracy: 0.6125
Epoch 40/1000
80/80 [==============================] - 0s 156us/sample - loss: 0.9351 - accuracy: 0.6000
Epoch 41/1000
80/80 [==============================] - 0s 143us/sample - loss: 0.9285 - accuracy: 0.6125
Epoch 42/1000
80/80 [==============================] - 0s 112us/sample - loss: 0.9313 - accuracy: 0.6375
Epoch 43/1000
```

```
80/80 [==============================] - 0s 135us/sample - loss: 0.9289 - accuracy: 0.6000
Epoch 44/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.9251 - accuracy: 0.6250
Epoch 45/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.9216 - accuracy: 0.6250
Epoch 46/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.9215 - accuracy: 0.6125
Epoch 47/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.9161 - accuracy: 0.6375
Epoch 48/1000
80/80 [==============================] - 0s 166us/sample - loss: 0.9127 - accuracy: 0.6125
Epoch 49/1000
80/80 [==============================] - 0s 159us/sample - loss: 0.9098 - accuracy: 0.5875
Epoch 50/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.9083 - accuracy: 0.6000
Epoch 51/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.9045 - accuracy: 0.6000
Epoch 52/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.9053 - accuracy: 0.6250
Epoch 53/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.9009 - accuracy: 0.6125
Epoch 54/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.9020 - accuracy: 0.6250
Epoch 55/1000
80/80 [==============================] - 0s 166us/sample - loss: 0.8946 - accuracy: 0.6625
Epoch 56/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.8911 - accuracy: 0.6375
Epoch 57/1000
80/80 [==============================] - 0s 172us/sample - loss: 0.8891 - accuracy: 0.6500
Epoch 58/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.8886 - accuracy: 0.6500
Epoch 59/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.8865 - accuracy: 0.6500
Epoch 60/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.8827 - accuracy: 0.6375
Epoch 61/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.8848 - accuracy: 0.6500
Epoch 62/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.8797 - accuracy: 0.6500
Epoch 63/1000
80/80 [==============================] - 0s 152us/sample - loss: 0.8775 - accuracy: 0.6250
Epoch 64/1000
80/80 [==============================] - 0s 198us/sample - loss: 0.8744 - accuracy: 0.6250
```

```
Epoch 65/1000
80/80 [==============================] - 0s 181us/sample - loss: 0.8724 - accuracy: 0.6250
Epoch 66/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.8681 - accuracy: 0.6375
Epoch 67/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.8712 - accuracy: 0.6500
Epoch 68/1000
80/80 [==============================] - 0s 143us/sample - loss: 0.8687 - accuracy: 0.6750
Epoch 69/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.8655 - accuracy: 0.6625
Epoch 70/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.8612 - accuracy: 0.6500
Epoch 71/1000
80/80 [==============================] - 0s 110us/sample - loss: 0.8600 - accuracy: 0.6500
Epoch 72/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.8585 - accuracy: 0.6625
Epoch 73/1000
80/80 [==============================] - 0s 112us/sample - loss: 0.8565 - accuracy: 0.6250
Epoch 74/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.8537 - accuracy: 0.6500
Epoch 75/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.8514 - accuracy: 0.6500
Epoch 76/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.8491 - accuracy: 0.6875
Epoch 77/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.8473 - accuracy: 0.6875
Epoch 78/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.8518 - accuracy: 0.6500
Epoch 79/1000
80/80 [==============================] - 0s 143us/sample - loss: 0.8496 - accuracy: 0.6500
Epoch 80/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.8407 - accuracy: 0.7000
Epoch 81/1000
80/80 [==============================] - 0s 108us/sample - loss: 0.8402 - accuracy: 0.6500
Epoch 82/1000
80/80 [==============================] - 0s 110us/sample - loss: 0.8456 - accuracy: 0.6375
Epoch 83/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.8386 - accuracy: 0.6625
Epoch 84/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.8353 - accuracy: 0.6750
Epoch 85/1000
80/80 [==============================] - 0s 165us/sample - loss: 0.8319 - accuracy: 0.6750
Epoch 86/1000
```

```
80/80 [==============================] - 0s 212us/sample - loss: 0.8321 - accuracy: 0.6750
Epoch 87/1000
80/80 [==============================] - 0s 147us/sample - loss: 0.8326 - accuracy: 0.6625
Epoch 88/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.8289 - accuracy: 0.6875
Epoch 89/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.8283 - accuracy: 0.6875
Epoch 90/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.8285 - accuracy: 0.6875
Epoch 91/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.8242 - accuracy: 0.6750
Epoch 92/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.8210 - accuracy: 0.6750
Epoch 93/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.8191 - accuracy: 0.6750
Epoch 94/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.8242 - accuracy: 0.6500
Epoch 95/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.8278 - accuracy: 0.6250
Epoch 96/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.8182 - accuracy: 0.6625
Epoch 97/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.8137 - accuracy: 0.6750
Epoch 98/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.8098 - accuracy: 0.6875
Epoch 99/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.8129 - accuracy: 0.6750
Epoch 100/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.8144 - accuracy: 0.6875
Epoch 101/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.8093 - accuracy: 0.7000
Epoch 102/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.8062 - accuracy: 0.6875
Epoch 103/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.8051 - accuracy: 0.6750
Epoch 104/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.8024 - accuracy: 0.6750
Epoch 105/1000
80/80 [==============================] - 0s 190us/sample - loss: 0.8047 - accuracy: 0.6875
Epoch 106/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.8030 - accuracy: 0.7000
Epoch 107/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.7988 - accuracy: 0.6750
```

```
Epoch 108/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.8012 - accuracy: 0.6750
Epoch 109/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.7991 - accuracy: 0.6750
Epoch 110/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.8031 - accuracy: 0.6875
Epoch 111/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.8019 - accuracy: 0.6875
Epoch 112/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.7965 - accuracy: 0.7000
Epoch 113/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.7917 - accuracy: 0.7000
Epoch 114/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.7867 - accuracy: 0.7000
Epoch 115/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.7870 - accuracy: 0.6875
Epoch 116/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.7870 - accuracy: 0.6750
Epoch 117/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.7909 - accuracy: 0.6750
Epoch 118/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.7840 - accuracy: 0.6750
Epoch 119/1000
80/80 [==============================] - 0s 158us/sample - loss: 0.7850 - accuracy: 0.6750
Epoch 120/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.7836 - accuracy: 0.6500
Epoch 121/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.7802 - accuracy: 0.6625
Epoch 122/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.7800 - accuracy: 0.7000
Epoch 123/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.7819 - accuracy: 0.7000
Epoch 124/1000
80/80 [==============================] - 0s 143us/sample - loss: 0.7782 - accuracy: 0.6875
Epoch 125/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.7747 - accuracy: 0.7000
Epoch 126/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.7762 - accuracy: 0.6750
Epoch 127/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.7740 - accuracy: 0.6750
Epoch 128/1000
80/80 [==============================] - 0s 159us/sample - loss: 0.7731 - accuracy: 0.7125
Epoch 129/1000
```

```
80/80 [==============================] - 0s 128us/sample - loss: 0.7725 - accuracy: 0.6750
Epoch 130/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.7730 - accuracy: 0.6750
Epoch 131/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.7707 - accuracy: 0.6750
Epoch 132/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.7696 - accuracy: 0.6875
Epoch 133/1000
80/80 [==============================] - 0s 160us/sample - loss: 0.7687 - accuracy: 0.7000
Epoch 134/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.7667 - accuracy: 0.6625
Epoch 135/1000
80/80 [==============================] - 0s 160us/sample - loss: 0.7712 - accuracy: 0.6500
Epoch 136/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.7678 - accuracy: 0.6500
Epoch 137/1000
80/80 [==============================] - 0s 158us/sample - loss: 0.7638 - accuracy: 0.6750
Epoch 138/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.7576 - accuracy: 0.7000
Epoch 139/1000
80/80 [==============================] - 0s 327us/sample - loss: 0.7629 - accuracy: 0.7000
Epoch 140/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.7672 - accuracy: 0.6875
Epoch 141/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.7622 - accuracy: 0.6750
Epoch 142/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.7563 - accuracy: 0.6875
Epoch 143/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.7565 - accuracy: 0.6875
Epoch 144/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.7594 - accuracy: 0.6875
Epoch 145/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.7575 - accuracy: 0.7000
Epoch 146/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.7539 - accuracy: 0.6875
Epoch 147/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.7538 - accuracy: 0.6625
Epoch 148/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.7542 - accuracy: 0.6875
Epoch 149/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.7528 - accuracy: 0.6750
Epoch 150/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.7505 - accuracy: 0.7000
```

```
Epoch 151/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.7529 - accuracy: 0.6875
Epoch 152/1000
80/80 [==============================] - 0s 165us/sample - loss: 0.7539 - accuracy: 0.6875
Epoch 153/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.7471 - accuracy: 0.7000
Epoch 154/1000
80/80 [==============================] - 0s 107us/sample - loss: 0.7444 - accuracy: 0.7000
Epoch 155/1000
80/80 [==============================] - 0s 268us/sample - loss: 0.7512 - accuracy: 0.6750
Epoch 156/1000
80/80 [==============================] - 0s 196us/sample - loss: 0.7507 - accuracy: 0.6500
Epoch 157/1000
80/80 [==============================] - 0s 203us/sample - loss: 0.7480 - accuracy: 0.6625
Epoch 158/1000
80/80 [==============================] - 0s 153us/sample - loss: 0.7464 - accuracy: 0.6750
Epoch 159/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.7408 - accuracy: 0.7000
Epoch 160/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.7481 - accuracy: 0.6750
Epoch 161/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.7425 - accuracy: 0.6750
Epoch 162/1000
80/80 [==============================] - 0s 176us/sample - loss: 0.7401 - accuracy: 0.6625
Epoch 163/1000
80/80 [==============================] - 0s 183us/sample - loss: 0.7377 - accuracy: 0.6750
Epoch 164/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.7400 - accuracy: 0.7000
Epoch 165/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.7468 - accuracy: 0.6875
Epoch 166/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.7457 - accuracy: 0.7000
Epoch 167/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.7375 - accuracy: 0.7000
Epoch 168/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.7346 - accuracy: 0.7000
Epoch 169/1000
80/80 [==============================] - 0s 161us/sample - loss: 0.7330 - accuracy: 0.7125
Epoch 170/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.7394 - accuracy: 0.6750
Epoch 171/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.7354 - accuracy: 0.7000
Epoch 172/1000
```

```
80/80 [==============================] - 0s 154us/sample - loss: 0.7347 - accuracy: 0.6875
Epoch 173/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.7297 - accuracy: 0.6750
Epoch 174/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.7348 - accuracy: 0.6875
Epoch 175/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.7326 - accuracy: 0.6875
Epoch 176/1000
80/80 [==============================] - 0s 166us/sample - loss: 0.7263 - accuracy: 0.7000
Epoch 177/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.7278 - accuracy: 0.7125
Epoch 178/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.7283 - accuracy: 0.7250
Epoch 179/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.7250 - accuracy: 0.7125
Epoch 180/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.7229 - accuracy: 0.6750
Epoch 181/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.7273 - accuracy: 0.6625
Epoch 182/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.7256 - accuracy: 0.6750
Epoch 183/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.7234 - accuracy: 0.6875
Epoch 184/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.7169 - accuracy: 0.7125
Epoch 185/1000
80/80 [==============================] - 0s 157us/sample - loss: 0.7202 - accuracy: 0.7125
Epoch 186/1000
80/80 [==============================] - 0s 157us/sample - loss: 0.7248 - accuracy: 0.7125
Epoch 187/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.7199 - accuracy: 0.6875
Epoch 188/1000
80/80 [==============================] - 0s 105us/sample - loss: 0.7176 - accuracy: 0.6875
Epoch 189/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.7132 - accuracy: 0.6750
Epoch 190/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.7184 - accuracy: 0.6875
Epoch 191/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.7121 - accuracy: 0.7000
Epoch 192/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.7199 - accuracy: 0.6500
Epoch 193/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.7158 - accuracy: 0.6500
```

```
Epoch 194/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.7094 - accuracy: 0.6750
Epoch 195/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.7094 - accuracy: 0.6875
Epoch 196/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.7122 - accuracy: 0.6750
Epoch 197/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.7073 - accuracy: 0.6750
Epoch 198/1000
80/80 [==============================] - 0s 166us/sample - loss: 0.7135 - accuracy: 0.7125
Epoch 199/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.7071 - accuracy: 0.7125
Epoch 200/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.7104 - accuracy: 0.6750
Epoch 201/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.7079 - accuracy: 0.6750
Epoch 202/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.7054 - accuracy: 0.6750
Epoch 203/1000
80/80 [==============================] - 0s 107us/sample - loss: 0.7025 - accuracy: 0.6750
Epoch 204/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.7014 - accuracy: 0.7000
Epoch 205/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.7065 - accuracy: 0.7000
Epoch 206/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.7077 - accuracy: 0.6625
Epoch 207/1000
80/80 [==============================] - 0s 103us/sample - loss: 0.7076 - accuracy: 0.6875
Epoch 208/1000
80/80 [==============================] - 0s 214us/sample - loss: 0.7025 - accuracy: 0.6500
Epoch 209/1000
80/80 [==============================] - 0s 184us/sample - loss: 0.7010 - accuracy: 0.6500
Epoch 210/1000
80/80 [==============================] - 0s 158us/sample - loss: 0.7055 - accuracy: 0.6875
Epoch 211/1000
80/80 [==============================] - 0s 160us/sample - loss: 0.7042 - accuracy: 0.6875
Epoch 212/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.7082 - accuracy: 0.6750
Epoch 213/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.6986 - accuracy: 0.6750
Epoch 214/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.7012 - accuracy: 0.6750
Epoch 215/1000
```

```
80/80 [==============================] - 0s 136us/sample - loss: 0.6970 - accuracy: 0.6750
Epoch 216/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.6962 - accuracy: 0.6875
Epoch 217/1000
80/80 [==============================] - 0s 162us/sample - loss: 0.6958 - accuracy: 0.7125
Epoch 218/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.6975 - accuracy: 0.6875
Epoch 219/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.6976 - accuracy: 0.6750
Epoch 220/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.6908 - accuracy: 0.6750
Epoch 221/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.6998 - accuracy: 0.6875
Epoch 222/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.7032 - accuracy: 0.6750
Epoch 223/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.6953 - accuracy: 0.6875
Epoch 224/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.6964 - accuracy: 0.6625
Epoch 225/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.6881 - accuracy: 0.6875
Epoch 226/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.6848 - accuracy: 0.7000
Epoch 227/1000
80/80 [==============================] - 0s 143us/sample - loss: 0.6892 - accuracy: 0.7000
Epoch 228/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.6873 - accuracy: 0.6750
Epoch 229/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.6862 - accuracy: 0.6625
Epoch 230/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.6870 - accuracy: 0.6750
Epoch 231/1000
80/80 [==============================] - 0s 177us/sample - loss: 0.6785 - accuracy: 0.6500
Epoch 232/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.6875 - accuracy: 0.6750
Epoch 233/1000
80/80 [==============================] - 0s 168us/sample - loss: 0.6958 - accuracy: 0.7125
Epoch 234/1000
80/80 [==============================] - 0s 157us/sample - loss: 0.6830 - accuracy: 0.7375
Epoch 235/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.6806 - accuracy: 0.6875
Epoch 236/1000
80/80 [==============================] - 0s 143us/sample - loss: 0.6857 - accuracy: 0.6625
```

```
Epoch 237/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.6834 - accuracy: 0.6500
Epoch 238/1000
80/80 [==============================] - 0s 162us/sample - loss: 0.6777 - accuracy: 0.6750
Epoch 239/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.6747 - accuracy: 0.7000
Epoch 240/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.6728 - accuracy: 0.7125
Epoch 241/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.6841 - accuracy: 0.7250
Epoch 242/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.6842 - accuracy: 0.7375
Epoch 243/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.6821 - accuracy: 0.7125
Epoch 244/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.6734 - accuracy: 0.6875
Epoch 245/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.6725 - accuracy: 0.6750
Epoch 246/1000
80/80 [==============================] - 0s 163us/sample - loss: 0.6786 - accuracy: 0.6625
Epoch 247/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.6697 - accuracy: 0.6875
Epoch 248/1000
80/80 [==============================] - 0s 161us/sample - loss: 0.6709 - accuracy: 0.7125
Epoch 249/1000
80/80 [==============================] - 0s 143us/sample - loss: 0.6755 - accuracy: 0.7250
Epoch 250/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.6708 - accuracy: 0.7250
Epoch 251/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.6716 - accuracy: 0.6750
Epoch 252/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.6703 - accuracy: 0.6875
Epoch 253/1000
80/80 [==============================] - 0s 104us/sample - loss: 0.6660 - accuracy: 0.6875
Epoch 254/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.6644 - accuracy: 0.6875
Epoch 255/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.6725 - accuracy: 0.7000
Epoch 256/1000
80/80 [==============================] - 0s 153us/sample - loss: 0.6688 - accuracy: 0.6875
Epoch 257/1000
80/80 [==============================] - 0s 192us/sample - loss: 0.6632 - accuracy: 0.7000
Epoch 258/1000
```

```
80/80 [==============================] - 0s 126us/sample - loss: 0.6691 - accuracy: 0.6625
Epoch 259/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.6629 - accuracy: 0.6750
Epoch 260/1000
80/80 [==============================] - 0s 162us/sample - loss: 0.6579 - accuracy: 0.6875
Epoch 261/1000
80/80 [==============================] - 0s 165us/sample - loss: 0.6664 - accuracy: 0.7250
Epoch 262/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.6653 - accuracy: 0.7250
Epoch 263/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.6683 - accuracy: 0.7000
Epoch 264/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.6595 - accuracy: 0.7000
Epoch 265/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.6623 - accuracy: 0.6750
Epoch 266/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.6626 - accuracy: 0.6750
Epoch 267/1000
80/80 [==============================] - 0s 105us/sample - loss: 0.6603 - accuracy: 0.7250
Epoch 268/1000
80/80 [==============================] - 0s 175us/sample - loss: 0.6611 - accuracy: 0.7375
Epoch 269/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.6542 - accuracy: 0.7250
Epoch 270/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.6547 - accuracy: 0.7250
Epoch 271/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.6602 - accuracy: 0.7125
Epoch 272/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.6616 - accuracy: 0.7125
Epoch 273/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.6532 - accuracy: 0.7250
Epoch 274/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.6570 - accuracy: 0.7000
Epoch 275/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.6551 - accuracy: 0.7000
Epoch 276/1000
80/80 [==============================] - 0s 156us/sample - loss: 0.6551 - accuracy: 0.7250
Epoch 277/1000
80/80 [==============================] - 0s 157us/sample - loss: 0.6516 - accuracy: 0.7125
Epoch 278/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.6499 - accuracy: 0.6875
Epoch 279/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.6512 - accuracy: 0.6750
```

```
Epoch 280/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.6520 - accuracy: 0.6625
Epoch 281/1000
80/80 [==============================] - 0s 153us/sample - loss: 0.6488 - accuracy: 0.6875
Epoch 282/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.6570 - accuracy: 0.7375
Epoch 283/1000
80/80 [==============================] - 0s 103us/sample - loss: 0.6486 - accuracy: 0.7250
Epoch 284/1000
80/80 [==============================] - 0s 266us/sample - loss: 0.6448 - accuracy: 0.7000
Epoch 285/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.6533 - accuracy: 0.7375
Epoch 286/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.6481 - accuracy: 0.6750
Epoch 287/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.6467 - accuracy: 0.7250
Epoch 288/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.6488 - accuracy: 0.7375
Epoch 289/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.6470 - accuracy: 0.7125
Epoch 290/1000
80/80 [==============================] - 0s 154us/sample - loss: 0.6454 - accuracy: 0.7000
Epoch 291/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.6439 - accuracy: 0.7000
Epoch 292/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.6433 - accuracy: 0.7000
Epoch 293/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.6486 - accuracy: 0.7125
Epoch 294/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.6468 - accuracy: 0.7500
Epoch 295/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.6461 - accuracy: 0.7125
Epoch 296/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.6438 - accuracy: 0.7250
Epoch 297/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.6384 - accuracy: 0.7375
Epoch 298/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.6368 - accuracy: 0.6875
Epoch 299/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.6415 - accuracy: 0.6750
Epoch 300/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.6446 - accuracy: 0.6875
Epoch 301/1000
```

```
80/80 [==============================] - 0s 142us/sample - loss: 0.6413 - accuracy: 0.6875
Epoch 302/1000
80/80 [==============================] - 0s 167us/sample - loss: 0.6375 - accuracy: 0.7000
Epoch 303/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.6348 - accuracy: 0.7250
Epoch 304/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.6378 - accuracy: 0.7125
Epoch 305/1000
80/80 [==============================] - 0s 210us/sample - loss: 0.6328 - accuracy: 0.7000
Epoch 306/1000
80/80 [==============================] - 0s 153us/sample - loss: 0.6305 - accuracy: 0.7250
Epoch 307/1000
80/80 [==============================] - 0s 160us/sample - loss: 0.6375 - accuracy: 0.7250
Epoch 308/1000
80/80 [==============================] - 0s 180us/sample - loss: 0.6359 - accuracy: 0.7250
Epoch 309/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.6358 - accuracy: 0.6750
Epoch 310/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.6356 - accuracy: 0.6875
Epoch 311/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.6363 - accuracy: 0.7125
Epoch 312/1000
80/80 [==============================] - 0s 112us/sample - loss: 0.6365 - accuracy: 0.7125
Epoch 313/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.6311 - accuracy: 0.7250
Epoch 314/1000
80/80 [==============================] - 0s 108us/sample - loss: 0.6340 - accuracy: 0.7125
Epoch 315/1000
80/80 [==============================] - 0s 112us/sample - loss: 0.6319 - accuracy: 0.6750
Epoch 316/1000
80/80 [==============================] - 0s 108us/sample - loss: 0.6293 - accuracy: 0.7000
Epoch 317/1000
80/80 [==============================] - 0s 195us/sample - loss: 0.6311 - accuracy: 0.7500
Epoch 318/1000
80/80 [==============================] - 0s 105us/sample - loss: 0.6262 - accuracy: 0.7250
Epoch 319/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.6231 - accuracy: 0.7000
Epoch 320/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.6331 - accuracy: 0.7000
Epoch 321/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.6270 - accuracy: 0.7000
Epoch 322/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.6253 - accuracy: 0.7000
```

```
Epoch 323/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.6210 - accuracy: 0.7000
Epoch 324/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.6233 - accuracy: 0.7250
Epoch 325/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.6228 - accuracy: 0.7125
Epoch 326/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.6162 - accuracy: 0.7125
Epoch 327/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.6265 - accuracy: 0.7250
Epoch 328/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.6268 - accuracy: 0.7375
Epoch 329/1000
80/80 [==============================] - 0s 153us/sample - loss: 0.6233 - accuracy: 0.7500
Epoch 330/1000
80/80 [==============================] - 0s 179us/sample - loss: 0.6159 - accuracy: 0.7500
Epoch 331/1000
80/80 [==============================] - 0s 165us/sample - loss: 0.6222 - accuracy: 0.7125
Epoch 332/1000
80/80 [==============================] - 0s 212us/sample - loss: 0.6200 - accuracy: 0.7125
Epoch 333/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.6162 - accuracy: 0.7250
Epoch 334/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.6165 - accuracy: 0.7375
Epoch 335/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.6245 - accuracy: 0.7250
Epoch 336/1000
80/80 [==============================] - 0s 106us/sample - loss: 0.6203 - accuracy: 0.7250
Epoch 337/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.6175 - accuracy: 0.7250
Epoch 338/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.6186 - accuracy: 0.7250
Epoch 339/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.6202 - accuracy: 0.7375
Epoch 340/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.6201 - accuracy: 0.7500
Epoch 341/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.6174 - accuracy: 0.7750
Epoch 342/1000
80/80 [==============================] - 0s 110us/sample - loss: 0.6228 - accuracy: 0.7000
Epoch 343/1000
80/80 [==============================] - 0s 100us/sample - loss: 0.6150 - accuracy: 0.6875
Epoch 344/1000
```

```
80/80 [==============================] - 0s 175us/sample - loss: 0.6125 - accuracy: 0.7250
Epoch 345/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.6094 - accuracy: 0.7250
Epoch 346/1000
80/80 [==============================] - 0s 152us/sample - loss: 0.6160 - accuracy: 0.7625
Epoch 347/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.6093 - accuracy: 0.7375
Epoch 348/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.6112 - accuracy: 0.7000
Epoch 349/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.6123 - accuracy: 0.7375
Epoch 350/1000
80/80 [==============================] - 0s 112us/sample - loss: 0.6122 - accuracy: 0.7625
Epoch 351/1000
80/80 [==============================] - 0s 143us/sample - loss: 0.6123 - accuracy: 0.7625
Epoch 352/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.6127 - accuracy: 0.7500
Epoch 353/1000
80/80 [==============================] - 0s 207us/sample - loss: 0.6091 - accuracy: 0.7375
Epoch 354/1000
80/80 [==============================] - 0s 186us/sample - loss: 0.6111 - accuracy: 0.7250
Epoch 355/1000
80/80 [==============================] - 0s 227us/sample - loss: 0.6103 - accuracy: 0.6750
Epoch 356/1000
80/80 [==============================] - 0s 186us/sample - loss: 0.6087 - accuracy: 0.6875
Epoch 357/1000
80/80 [==============================] - 0s 152us/sample - loss: 0.6053 - accuracy: 0.7250
Epoch 358/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.6108 - accuracy: 0.7500
Epoch 359/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.6054 - accuracy: 0.7500
Epoch 360/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.6032 - accuracy: 0.7125
Epoch 361/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.6021 - accuracy: 0.7000
Epoch 362/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.6094 - accuracy: 0.6875
Epoch 363/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.6067 - accuracy: 0.7000
Epoch 364/1000
80/80 [==============================] - 0s 106us/sample - loss: 0.5992 - accuracy: 0.7625
Epoch 365/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.6011 - accuracy: 0.7375
```

```
Epoch 366/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.5990 - accuracy: 0.7375
Epoch 367/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.5999 - accuracy: 0.7500
Epoch 368/1000
80/80 [==============================] - 0s 158us/sample - loss: 0.5980 - accuracy: 0.7375
Epoch 369/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.6023 - accuracy: 0.7125
Epoch 370/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.6013 - accuracy: 0.7000
Epoch 371/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.5960 - accuracy: 0.7125
Epoch 372/1000
80/80 [==============================] - 0s 147us/sample - loss: 0.6038 - accuracy: 0.7500
Epoch 373/1000
80/80 [==============================] - 0s 143us/sample - loss: 0.6004 - accuracy: 0.7750
Epoch 374/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.6000 - accuracy: 0.7375
Epoch 375/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.5994 - accuracy: 0.7625
Epoch 376/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.6051 - accuracy: 0.7750
Epoch 377/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.5969 - accuracy: 0.7250
Epoch 378/1000
80/80 [==============================] - 0s 196us/sample - loss: 0.5894 - accuracy: 0.7500
Epoch 379/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.5998 - accuracy: 0.7500
Epoch 380/1000
80/80 [==============================] - 0s 165us/sample - loss: 0.6005 - accuracy: 0.7750
Epoch 381/1000
80/80 [==============================] - 0s 155us/sample - loss: 0.5958 - accuracy: 0.7500
Epoch 382/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.5891 - accuracy: 0.7625
Epoch 383/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.5917 - accuracy: 0.7500
Epoch 384/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.5928 - accuracy: 0.7250
Epoch 385/1000
80/80 [==============================] - 0s 108us/sample - loss: 0.5905 - accuracy: 0.7375
Epoch 386/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.5903 - accuracy: 0.7500
Epoch 387/1000
```

```
80/80 [==============================] - 0s 142us/sample - loss: 0.5871 - accuracy: 0.7875
Epoch 388/1000
80/80 [==============================] - 0s 152us/sample - loss: 0.5888 - accuracy: 0.7875
Epoch 389/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.5946 - accuracy: 0.7750
Epoch 390/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.5885 - accuracy: 0.8000
Epoch 391/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.5988 - accuracy: 0.7500
Epoch 392/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.5961 - accuracy: 0.7375
Epoch 393/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.5871 - accuracy: 0.7500
Epoch 394/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.5898 - accuracy: 0.7250
Epoch 395/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.5834 - accuracy: 0.7750
Epoch 396/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.5886 - accuracy: 0.7500
Epoch 397/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.5870 - accuracy: 0.7875
Epoch 398/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.5873 - accuracy: 0.7750
Epoch 399/1000
80/80 [==============================] - 0s 178us/sample - loss: 0.5883 - accuracy: 0.7625
Epoch 400/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.5875 - accuracy: 0.7375
Epoch 401/1000
80/80 [==============================] - 0s 171us/sample - loss: 0.5828 - accuracy: 0.7500
Epoch 402/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.5895 - accuracy: 0.7125
Epoch 403/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.5827 - accuracy: 0.7000
Epoch 404/1000
80/80 [==============================] - 0s 163us/sample - loss: 0.5821 - accuracy: 0.7500
Epoch 405/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.5850 - accuracy: 0.7750
Epoch 406/1000
80/80 [==============================] - 0s 162us/sample - loss: 0.5895 - accuracy: 0.7750
Epoch 407/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.5894 - accuracy: 0.7625
Epoch 408/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.5771 - accuracy: 0.7375
```

```
Epoch 409/1000
80/80 [==============================] - 0s 97us/sample - loss: 0.5812 - accuracy: 0.7500
Epoch 410/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.5795 - accuracy: 0.7750
Epoch 411/1000
80/80 [==============================] - 0s 159us/sample - loss: 0.5784 - accuracy: 0.7625
Epoch 412/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.5905 - accuracy: 0.7375
Epoch 413/1000
80/80 [==============================] - 0s 212us/sample - loss: 0.5817 - accuracy: 0.7250
Epoch 414/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.5752 - accuracy: 0.7250
Epoch 415/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.5742 - accuracy: 0.7625
Epoch 416/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.5754 - accuracy: 0.7375
Epoch 417/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.5757 - accuracy: 0.7375
Epoch 418/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.5783 - accuracy: 0.7625
Epoch 419/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.5714 - accuracy: 0.7750
Epoch 420/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.5784 - accuracy: 0.7750
Epoch 421/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.5728 - accuracy: 0.7875
Epoch 422/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.5720 - accuracy: 0.7625
Epoch 423/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.5792 - accuracy: 0.7625
Epoch 424/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.5782 - accuracy: 0.7625
Epoch 425/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.5705 - accuracy: 0.7625
Epoch 426/1000
80/80 [==============================] - 0s 110us/sample - loss: 0.5717 - accuracy: 0.7750
Epoch 427/1000
80/80 [==============================] - 0s 108us/sample - loss: 0.5759 - accuracy: 0.7500
Epoch 428/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.5727 - accuracy: 0.7500
Epoch 429/1000
80/80 [==============================] - 0s 225us/sample - loss: 0.5670 - accuracy: 0.7500
Epoch 430/1000
```

```
80/80 [==============================] - 0s 159us/sample - loss: 0.5702 - accuracy: 0.7625
Epoch 431/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.5770 - accuracy: 0.7500
Epoch 432/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.5705 - accuracy: 0.7625
Epoch 433/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.5702 - accuracy: 0.7625
Epoch 434/1000
80/80 [==============================] - 0s 160us/sample - loss: 0.5671 - accuracy: 0.7625
Epoch 435/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.5744 - accuracy: 0.7500
Epoch 436/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.5688 - accuracy: 0.8000
Epoch 437/1000
80/80 [==============================] - 0s 104us/sample - loss: 0.5698 - accuracy: 0.8000
Epoch 438/1000
80/80 [==============================] - 0s 156us/sample - loss: 0.5735 - accuracy: 0.7625
Epoch 439/1000
80/80 [==============================] - 0s 105us/sample - loss: 0.5765 - accuracy: 0.7375
Epoch 440/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.5656 - accuracy: 0.7250
Epoch 441/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.5633 - accuracy: 0.7500
Epoch 442/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.5665 - accuracy: 0.7750
Epoch 443/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.5687 - accuracy: 0.8000
Epoch 444/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.5660 - accuracy: 0.8000
Epoch 445/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.5705 - accuracy: 0.7625
Epoch 446/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.5704 - accuracy: 0.7375
Epoch 447/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.5691 - accuracy: 0.7125
Epoch 448/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.5697 - accuracy: 0.6750
Epoch 449/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.5698 - accuracy: 0.7250
Epoch 450/1000
80/80 [==============================] - 0s 167us/sample - loss: 0.5655 - accuracy: 0.7625
Epoch 451/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.5589 - accuracy: 0.7750
```

```
Epoch 452/1000
80/80 [==============================] - 0s 205us/sample - loss: 0.5644 - accuracy: 0.7625
Epoch 453/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.5621 - accuracy: 0.8000
Epoch 454/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.5602 - accuracy: 0.8000
Epoch 455/1000
80/80 [==============================] - 0s 108us/sample - loss: 0.5580 - accuracy: 0.7875
Epoch 456/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.5594 - accuracy: 0.7500
Epoch 457/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.5589 - accuracy: 0.7375
Epoch 458/1000
80/80 [==============================] - 0s 168us/sample - loss: 0.5653 - accuracy: 0.7500
Epoch 459/1000
80/80 [==============================] - 0s 89us/sample - loss: 0.5588 - accuracy: 0.8000
Epoch 460/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.5561 - accuracy: 0.8000
Epoch 461/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.5572 - accuracy: 0.7875
Epoch 462/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.5574 - accuracy: 0.7750
Epoch 463/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.5566 - accuracy: 0.7500
Epoch 464/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.5565 - accuracy: 0.7625
Epoch 465/1000
80/80 [==============================] - 0s 106us/sample - loss: 0.5558 - accuracy: 0.7875
Epoch 466/1000
80/80 [==============================] - 0s 110us/sample - loss: 0.5514 - accuracy: 0.8000
Epoch 467/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.5661 - accuracy: 0.8000
Epoch 468/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.5594 - accuracy: 0.8000
Epoch 469/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.5572 - accuracy: 0.7500
Epoch 470/1000
80/80 [==============================] - 0s 170us/sample - loss: 0.5570 - accuracy: 0.7500
Epoch 471/1000
80/80 [==============================] - 0s 154us/sample - loss: 0.5581 - accuracy: 0.7500
Epoch 472/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.5549 - accuracy: 0.8125
Epoch 473/1000
```

```
80/80 [==============================] - 0s 158us/sample - loss: 0.5536 - accuracy: 0.8000
Epoch 474/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.5532 - accuracy: 0.7875
Epoch 475/1000
80/80 [==============================] - 0s 106us/sample - loss: 0.5517 - accuracy: 0.7750
Epoch 476/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.5536 - accuracy: 0.7875
Epoch 477/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.5473 - accuracy: 0.8000
Epoch 478/1000
80/80 [==============================] - 0s 96us/sample - loss: 0.5558 - accuracy: 0.7750
Epoch 479/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.5505 - accuracy: 0.7875
Epoch 480/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.5501 - accuracy: 0.7750
Epoch 481/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.5579 - accuracy: 0.7625
Epoch 482/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.5540 - accuracy: 0.7750
Epoch 483/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.5516 - accuracy: 0.8000
Epoch 484/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.5547 - accuracy: 0.7625
Epoch 485/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.5551 - accuracy: 0.7375
Epoch 486/1000
80/80 [==============================] - 0s 182us/sample - loss: 0.5460 - accuracy: 0.7875
Epoch 487/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.5456 - accuracy: 0.7750
Epoch 488/1000
80/80 [==============================] - 0s 164us/sample - loss: 0.5460 - accuracy: 0.7875
Epoch 489/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.5440 - accuracy: 0.7875
Epoch 490/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.5461 - accuracy: 0.7750
Epoch 491/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.5443 - accuracy: 0.7750
Epoch 492/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.5444 - accuracy: 0.7500
Epoch 493/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.5453 - accuracy: 0.7750
Epoch 494/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.5404 - accuracy: 0.7750
```

```
Epoch 495/1000
80/80 [==============================] - 0s 158us/sample - loss: 0.5433 - accuracy: 0.7875
Epoch 496/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.5487 - accuracy: 0.8125
Epoch 497/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.5465 - accuracy: 0.7875
Epoch 498/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.5376 - accuracy: 0.7750
Epoch 499/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.5437 - accuracy: 0.7500
Epoch 500/1000
80/80 [==============================] - 0s 172us/sample - loss: 0.5445 - accuracy: 0.7500
Epoch 501/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.5435 - accuracy: 0.7875
Epoch 502/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.5481 - accuracy: 0.7625
Epoch 503/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.5529 - accuracy: 0.7875
Epoch 504/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.5390 - accuracy: 0.8250
Epoch 505/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.5445 - accuracy: 0.7875
Epoch 506/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.5376 - accuracy: 0.7875
Epoch 507/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.5362 - accuracy: 0.7875
Epoch 508/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.5384 - accuracy: 0.8000
Epoch 509/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.5398 - accuracy: 0.8000
Epoch 510/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.5371 - accuracy: 0.7875
Epoch 511/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.5399 - accuracy: 0.8000
Epoch 512/1000
80/80 [==============================] - 0s 199us/sample - loss: 0.5383 - accuracy: 0.8000
Epoch 513/1000
80/80 [==============================] - 0s 104us/sample - loss: 0.5386 - accuracy: 0.7875
Epoch 514/1000
80/80 [==============================] - 0s 155us/sample - loss: 0.5357 - accuracy: 0.7875
Epoch 515/1000
80/80 [==============================] - 0s 156us/sample - loss: 0.5350 - accuracy: 0.7875
Epoch 516/1000
```

```
80/80 [==============================] - 0s 127us/sample - loss: 0.5391 - accuracy: 0.7625
Epoch 517/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.5359 - accuracy: 0.7875
Epoch 518/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.5313 - accuracy: 0.7875
Epoch 519/1000
80/80 [==============================] - 0s 164us/sample - loss: 0.5328 - accuracy: 0.8000
Epoch 520/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.5324 - accuracy: 0.8000
Epoch 521/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.5303 - accuracy: 0.8125
Epoch 522/1000
80/80 [==============================] - 0s 169us/sample - loss: 0.5308 - accuracy: 0.8000
Epoch 523/1000
80/80 [==============================] - 0s 176us/sample - loss: 0.5339 - accuracy: 0.8000
Epoch 524/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.5332 - accuracy: 0.8000
Epoch 525/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.5360 - accuracy: 0.7750
Epoch 526/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.5365 - accuracy: 0.7875
Epoch 527/1000
80/80 [==============================] - 0s 172us/sample - loss: 0.5290 - accuracy: 0.7750
Epoch 528/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.5303 - accuracy: 0.7625
Epoch 529/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.5328 - accuracy: 0.7875
Epoch 530/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.5291 - accuracy: 0.7875
Epoch 531/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.5307 - accuracy: 0.7875
Epoch 532/1000
80/80 [==============================] - 0s 99us/sample - loss: 0.5281 - accuracy: 0.7875
Epoch 533/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.5299 - accuracy: 0.7875
Epoch 534/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.5280 - accuracy: 0.7875
Epoch 535/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.5246 - accuracy: 0.8000
Epoch 536/1000
80/80 [==============================] - 0s 153us/sample - loss: 0.5313 - accuracy: 0.7875
Epoch 537/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.5279 - accuracy: 0.8000
```

```
Epoch 538/1000
80/80 [==============================] - 0s 160us/sample - loss: 0.5283 - accuracy: 0.7875
Epoch 539/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.5265 - accuracy: 0.7875
Epoch 540/1000
80/80 [==============================] - 0s 160us/sample - loss: 0.5303 - accuracy: 0.8125
Epoch 541/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.5342 - accuracy: 0.7750
Epoch 542/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.5288 - accuracy: 0.7625
Epoch 543/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.5304 - accuracy: 0.8000
Epoch 544/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.5243 - accuracy: 0.8000
Epoch 545/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.5283 - accuracy: 0.7875
Epoch 546/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.5296 - accuracy: 0.7750
Epoch 547/1000
80/80 [==============================] - 0s 104us/sample - loss: 0.5250 - accuracy: 0.7750
Epoch 548/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.5252 - accuracy: 0.8000
Epoch 549/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.5227 - accuracy: 0.8000
Epoch 550/1000
80/80 [==============================] - 0s 153us/sample - loss: 0.5254 - accuracy: 0.7875
Epoch 551/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.5272 - accuracy: 0.7750
Epoch 552/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.5305 - accuracy: 0.7750
Epoch 553/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.5194 - accuracy: 0.7750
Epoch 554/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.5238 - accuracy: 0.7750
Epoch 555/1000
80/80 [==============================] - 0s 162us/sample - loss: 0.5326 - accuracy: 0.7750
Epoch 556/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.5236 - accuracy: 0.7750
Epoch 557/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.5240 - accuracy: 0.7875
Epoch 558/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.5257 - accuracy: 0.8250
Epoch 559/1000
```

```
80/80 [==============================] - 0s 117us/sample - loss: 0.5205 - accuracy: 0.8375
Epoch 560/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.5204 - accuracy: 0.8000
Epoch 561/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.5266 - accuracy: 0.7875
Epoch 562/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.5278 - accuracy: 0.7500
Epoch 563/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.5247 - accuracy: 0.7875
Epoch 564/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.5176 - accuracy: 0.8000
Epoch 565/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.5192 - accuracy: 0.8000
Epoch 566/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.5157 - accuracy: 0.8000
Epoch 567/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.5162 - accuracy: 0.8000
Epoch 568/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.5170 - accuracy: 0.7875
Epoch 569/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.5177 - accuracy: 0.7875
Epoch 570/1000
80/80 [==============================] - 0s 198us/sample - loss: 0.5165 - accuracy: 0.7875
Epoch 571/1000
80/80 [==============================] - 0s 165us/sample - loss: 0.5159 - accuracy: 0.8000
Epoch 572/1000
80/80 [==============================] - 0s 182us/sample - loss: 0.5167 - accuracy: 0.8125
Epoch 573/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.5211 - accuracy: 0.8125
Epoch 574/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.5182 - accuracy: 0.8125
Epoch 575/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.5174 - accuracy: 0.7750
Epoch 576/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.5159 - accuracy: 0.8000
Epoch 577/1000
80/80 [==============================] - 0s 188us/sample - loss: 0.5186 - accuracy: 0.7750
Epoch 578/1000
80/80 [==============================] - 0s 166us/sample - loss: 0.5161 - accuracy: 0.8000
Epoch 579/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.5237 - accuracy: 0.8125
Epoch 580/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.5179 - accuracy: 0.8125
```

```
Epoch 581/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.5104 - accuracy: 0.7875
Epoch 582/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.5170 - accuracy: 0.7875
Epoch 583/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.5178 - accuracy: 0.7750
Epoch 584/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.5163 - accuracy: 0.8000
Epoch 585/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.5092 - accuracy: 0.7875
Epoch 586/1000
80/80 [==============================] - 0s 106us/sample - loss: 0.5160 - accuracy: 0.8250
Epoch 587/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.5163 - accuracy: 0.7875
Epoch 588/1000
80/80 [==============================] - 0s 158us/sample - loss: 0.5176 - accuracy: 0.7750
Epoch 589/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.5115 - accuracy: 0.7750
Epoch 590/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.5161 - accuracy: 0.7750
Epoch 591/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.5126 - accuracy: 0.7875
Epoch 592/1000
80/80 [==============================] - 0s 163us/sample - loss: 0.5090 - accuracy: 0.7875
Epoch 593/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.5213 - accuracy: 0.8000
Epoch 594/1000
80/80 [==============================] - 0s 206us/sample - loss: 0.5254 - accuracy: 0.8000
Epoch 595/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.5103 - accuracy: 0.8000
Epoch 596/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.5105 - accuracy: 0.8000
Epoch 597/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.5093 - accuracy: 0.8000
Epoch 598/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.5089 - accuracy: 0.7750
Epoch 599/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.5054 - accuracy: 0.8125
Epoch 600/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.5165 - accuracy: 0.7875
Epoch 601/1000
80/80 [==============================] - 0s 170us/sample - loss: 0.5137 - accuracy: 0.8000
Epoch 602/1000
```

```
80/80 [==============================] - 0s 167us/sample - loss: 0.5049 - accuracy: 0.8125
Epoch 603/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.5126 - accuracy: 0.7875
Epoch 604/1000
80/80 [==============================] - 0s 108us/sample - loss: 0.5111 - accuracy: 0.7875
Epoch 605/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.5098 - accuracy: 0.7875
Epoch 606/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.5067 - accuracy: 0.8125
Epoch 607/1000
80/80 [==============================] - 0s 110us/sample - loss: 0.5072 - accuracy: 0.7750
Epoch 608/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.5043 - accuracy: 0.7875
Epoch 609/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.5131 - accuracy: 0.8000
Epoch 610/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.5049 - accuracy: 0.8000
Epoch 611/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.5006 - accuracy: 0.8125
Epoch 612/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.5096 - accuracy: 0.7875
Epoch 613/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.5061 - accuracy: 0.7875
Epoch 614/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.5031 - accuracy: 0.8125
Epoch 615/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.5049 - accuracy: 0.7875
Epoch 616/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.5034 - accuracy: 0.8000
Epoch 617/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.5014 - accuracy: 0.8000
Epoch 618/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.5046 - accuracy: 0.8125
Epoch 619/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.5035 - accuracy: 0.7750
Epoch 620/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.4996 - accuracy: 0.7875
Epoch 621/1000
80/80 [==============================] - 0s 112us/sample - loss: 0.5057 - accuracy: 0.7875
Epoch 622/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.5036 - accuracy: 0.8000
Epoch 623/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.5041 - accuracy: 0.8000
```

```
Epoch 624/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.5002 - accuracy: 0.8000
Epoch 625/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.4962 - accuracy: 0.8000
Epoch 626/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.5054 - accuracy: 0.7875
Epoch 627/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.5033 - accuracy: 0.8000
Epoch 628/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.5063 - accuracy: 0.7875
Epoch 629/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4977 - accuracy: 0.8000
Epoch 630/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.4958 - accuracy: 0.8250
Epoch 631/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.4973 - accuracy: 0.8125
Epoch 632/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.4943 - accuracy: 0.8125
Epoch 633/1000
80/80 [==============================] - 0s 175us/sample - loss: 0.4920 - accuracy: 0.8125
Epoch 634/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.4933 - accuracy: 0.8125
Epoch 635/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.4929 - accuracy: 0.8250
Epoch 636/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.4939 - accuracy: 0.8125
Epoch 637/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.4963 - accuracy: 0.8125
Epoch 638/1000
80/80 [==============================] - 0s 153us/sample - loss: 0.4978 - accuracy: 0.7750
Epoch 639/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.4943 - accuracy: 0.7875
Epoch 640/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4974 - accuracy: 0.8000
Epoch 641/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4936 - accuracy: 0.8000
Epoch 642/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.4904 - accuracy: 0.8125
Epoch 643/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.5030 - accuracy: 0.8000
Epoch 644/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.4939 - accuracy: 0.8250
Epoch 645/1000
```

```
80/80 [==============================] - 0s 171us/sample - loss: 0.4953 - accuracy: 0.8000
Epoch 646/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4905 - accuracy: 0.8000
Epoch 647/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4909 - accuracy: 0.8125
Epoch 648/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.4965 - accuracy: 0.8125
Epoch 649/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4972 - accuracy: 0.8125
Epoch 650/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4944 - accuracy: 0.8000
Epoch 651/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.4936 - accuracy: 0.7875
Epoch 652/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.4851 - accuracy: 0.8000
Epoch 653/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.5010 - accuracy: 0.8000
Epoch 654/1000
80/80 [==============================] - 0s 185us/sample - loss: 0.5000 - accuracy: 0.7875
Epoch 655/1000
80/80 [==============================] - 0s 218us/sample - loss: 0.5024 - accuracy: 0.7875
Epoch 656/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.4888 - accuracy: 0.8000
Epoch 657/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.4883 - accuracy: 0.7875
Epoch 658/1000
80/80 [==============================] - 0s 106us/sample - loss: 0.4916 - accuracy: 0.7875
Epoch 659/1000
80/80 [==============================] - 0s 154us/sample - loss: 0.4917 - accuracy: 0.8000
Epoch 660/1000
80/80 [==============================] - 0s 110us/sample - loss: 0.4867 - accuracy: 0.8125
Epoch 661/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4904 - accuracy: 0.7875
Epoch 662/1000
80/80 [==============================] - 0s 158us/sample - loss: 0.4890 - accuracy: 0.8125
Epoch 663/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4831 - accuracy: 0.8125
Epoch 664/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.4819 - accuracy: 0.8250
Epoch 665/1000
80/80 [==============================] - 0s 162us/sample - loss: 0.4854 - accuracy: 0.8000
Epoch 666/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.4991 - accuracy: 0.8250
```

```
Epoch 667/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.4853 - accuracy: 0.8000
Epoch 668/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4829 - accuracy: 0.8125
Epoch 669/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.4921 - accuracy: 0.8125
Epoch 670/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.4868 - accuracy: 0.7750
Epoch 671/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4837 - accuracy: 0.8000
Epoch 672/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.4988 - accuracy: 0.8000
Epoch 673/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.4822 - accuracy: 0.8250
Epoch 674/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.4968 - accuracy: 0.7875
Epoch 675/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4997 - accuracy: 0.7750
Epoch 676/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.4879 - accuracy: 0.8000
Epoch 677/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.4921 - accuracy: 0.8125
Epoch 678/1000
80/80 [==============================] - 0s 99us/sample - loss: 0.4878 - accuracy: 0.7875
Epoch 679/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.4836 - accuracy: 0.8125
Epoch 680/1000
80/80 [==============================] - 0s 154us/sample - loss: 0.4838 - accuracy: 0.7875
Epoch 681/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4840 - accuracy: 0.8000
Epoch 682/1000
80/80 [==============================] - 0s 154us/sample - loss: 0.4799 - accuracy: 0.8250
Epoch 683/1000
80/80 [==============================] - 0s 190us/sample - loss: 0.4804 - accuracy: 0.8125
Epoch 684/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.4819 - accuracy: 0.7875
Epoch 685/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4818 - accuracy: 0.8000
Epoch 686/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.4762 - accuracy: 0.8125
Epoch 687/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.4773 - accuracy: 0.8000
Epoch 688/1000
```

```
80/80 [==============================] - 0s 123us/sample - loss: 0.4828 - accuracy: 0.7875
Epoch 689/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.4755 - accuracy: 0.8000
Epoch 690/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.4774 - accuracy: 0.8375
Epoch 691/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.4824 - accuracy: 0.8125
Epoch 692/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.4826 - accuracy: 0.7875
Epoch 693/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.4768 - accuracy: 0.7875
Epoch 694/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.4737 - accuracy: 0.8500
Epoch 695/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.4775 - accuracy: 0.8250
Epoch 696/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.4817 - accuracy: 0.8125
Epoch 697/1000
80/80 [==============================] - 0s 156us/sample - loss: 0.4836 - accuracy: 0.7875
Epoch 698/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.4831 - accuracy: 0.7875
Epoch 699/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4795 - accuracy: 0.7875
Epoch 700/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.4742 - accuracy: 0.7750
Epoch 701/1000
80/80 [==============================] - 0s 205us/sample - loss: 0.4835 - accuracy: 0.8000
Epoch 702/1000
80/80 [==============================] - 0s 147us/sample - loss: 0.4810 - accuracy: 0.8375
Epoch 703/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.4745 - accuracy: 0.8125
Epoch 704/1000
80/80 [==============================] - 0s 161us/sample - loss: 0.4780 - accuracy: 0.8125
Epoch 705/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.4794 - accuracy: 0.8125
Epoch 706/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.4787 - accuracy: 0.8125
Epoch 707/1000
80/80 [==============================] - 0s 104us/sample - loss: 0.4771 - accuracy: 0.8375
Epoch 708/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.4762 - accuracy: 0.8250
Epoch 709/1000
80/80 [==============================] - 0s 107us/sample - loss: 0.4710 - accuracy: 0.8250
```

```
Epoch 710/1000
80/80 [==============================] - 0s 155us/sample - loss: 0.4732 - accuracy: 0.8000
Epoch 711/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.4715 - accuracy: 0.7875
Epoch 712/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.4735 - accuracy: 0.8125
Epoch 713/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.4776 - accuracy: 0.8000
Epoch 714/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.4709 - accuracy: 0.8250
Epoch 715/1000
80/80 [==============================] - 0s 107us/sample - loss: 0.4694 - accuracy: 0.8125
Epoch 716/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4760 - accuracy: 0.8125
Epoch 717/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.4717 - accuracy: 0.7875
Epoch 718/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4731 - accuracy: 0.7750
Epoch 719/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.4661 - accuracy: 0.8000
Epoch 720/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.4703 - accuracy: 0.8000
Epoch 721/1000
80/80 [==============================] - 0s 147us/sample - loss: 0.4702 - accuracy: 0.8125
Epoch 722/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4740 - accuracy: 0.7875
Epoch 723/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.4714 - accuracy: 0.8000
Epoch 724/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.4714 - accuracy: 0.8375
Epoch 725/1000
80/80 [==============================] - 0s 108us/sample - loss: 0.4693 - accuracy: 0.8000
Epoch 726/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.4654 - accuracy: 0.8125
Epoch 727/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.4707 - accuracy: 0.8125
Epoch 728/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4711 - accuracy: 0.8375
Epoch 729/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.4712 - accuracy: 0.7875
Epoch 730/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.4717 - accuracy: 0.7875
Epoch 731/1000
```

```
80/80 [==============================] - 0s 134us/sample - loss: 0.4694 - accuracy: 0.8000
Epoch 732/1000
80/80 [==============================] - 0s 203us/sample - loss: 0.4682 - accuracy: 0.8125
Epoch 733/1000
80/80 [==============================] - 0s 153us/sample - loss: 0.4685 - accuracy: 0.7875
Epoch 734/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.4626 - accuracy: 0.8125
Epoch 735/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.4624 - accuracy: 0.8125
Epoch 736/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4707 - accuracy: 0.8125
Epoch 737/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.4682 - accuracy: 0.7875
Epoch 738/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.4681 - accuracy: 0.7875
Epoch 739/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.4629 - accuracy: 0.7875
Epoch 740/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.4636 - accuracy: 0.8000
Epoch 741/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4701 - accuracy: 0.7875
Epoch 742/1000
80/80 [==============================] - 0s 99us/sample - loss: 0.4675 - accuracy: 0.8125
Epoch 743/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4611 - accuracy: 0.8500
Epoch 744/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4617 - accuracy: 0.8375
Epoch 745/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4656 - accuracy: 0.8125
Epoch 746/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.4700 - accuracy: 0.8375
Epoch 747/1000
80/80 [==============================] - 0s 94us/sample - loss: 0.4662 - accuracy: 0.8250
Epoch 748/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.4594 - accuracy: 0.8000
Epoch 749/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.4583 - accuracy: 0.8250
Epoch 750/1000
80/80 [==============================] - 0s 103us/sample - loss: 0.4660 - accuracy: 0.8250
Epoch 751/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.4640 - accuracy: 0.8500
Epoch 752/1000
80/80 [==============================] - 0s 108us/sample - loss: 0.4567 - accuracy: 0.7875
```

```
Epoch 753/1000
80/80 [==============================] - 0s 154us/sample - loss: 0.4702 - accuracy: 0.8125
Epoch 754/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4644 - accuracy: 0.8250
Epoch 755/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4664 - accuracy: 0.8375
Epoch 756/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4580 - accuracy: 0.8125
Epoch 757/1000
80/80 [==============================] - 0s 147us/sample - loss: 0.4571 - accuracy: 0.8250
Epoch 758/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4613 - accuracy: 0.8000
Epoch 759/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.4606 - accuracy: 0.8000
Epoch 760/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.4589 - accuracy: 0.8125
Epoch 761/1000
80/80 [==============================] - 0s 141us/sample - loss: 0.4580 - accuracy: 0.8375
Epoch 762/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.4587 - accuracy: 0.8250
Epoch 763/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.4577 - accuracy: 0.8500
Epoch 764/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4621 - accuracy: 0.8125
Epoch 765/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4587 - accuracy: 0.8125
Epoch 766/1000
80/80 [==============================] - 0s 156us/sample - loss: 0.4547 - accuracy: 0.8250
Epoch 767/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.4633 - accuracy: 0.8375
Epoch 768/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.4619 - accuracy: 0.8000
Epoch 769/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.4511 - accuracy: 0.8375
Epoch 770/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.4790 - accuracy: 0.7875
Epoch 771/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4727 - accuracy: 0.8375
Epoch 772/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.4606 - accuracy: 0.8000
Epoch 773/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.4547 - accuracy: 0.7875
Epoch 774/1000
```

```
80/80 [==============================] - 0s 98us/sample - loss: 0.4632 - accuracy: 0.8125
Epoch 775/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4537 - accuracy: 0.8250
Epoch 776/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.4583 - accuracy: 0.8000
Epoch 777/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.4558 - accuracy: 0.8250
Epoch 778/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4478 - accuracy: 0.8375
Epoch 779/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.4553 - accuracy: 0.8125
Epoch 780/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4576 - accuracy: 0.8250
Epoch 781/1000
80/80 [==============================] - 0s 102us/sample - loss: 0.4509 - accuracy: 0.8375
Epoch 782/1000
80/80 [==============================] - 0s 99us/sample - loss: 0.4599 - accuracy: 0.8250
Epoch 783/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.4549 - accuracy: 0.8375
Epoch 784/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4517 - accuracy: 0.8000
Epoch 785/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4528 - accuracy: 0.8125
Epoch 786/1000
80/80 [==============================] - 0s 165us/sample - loss: 0.4555 - accuracy: 0.8250
Epoch 787/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4577 - accuracy: 0.8125
Epoch 788/1000
80/80 [==============================] - 0s 154us/sample - loss: 0.4520 - accuracy: 0.8625
Epoch 789/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.4510 - accuracy: 0.8375
Epoch 790/1000
80/80 [==============================] - 0s 164us/sample - loss: 0.4512 - accuracy: 0.8125
Epoch 791/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.4482 - accuracy: 0.8125
Epoch 792/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4529 - accuracy: 0.8125
Epoch 793/1000
80/80 [==============================] - 0s 143us/sample - loss: 0.4552 - accuracy: 0.8125
Epoch 794/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.4574 - accuracy: 0.8000
Epoch 795/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.4527 - accuracy: 0.8250
```

```
Epoch 796/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.4480 - accuracy: 0.8125
Epoch 797/1000
80/80 [==============================] - 0s 110us/sample - loss: 0.4528 - accuracy: 0.8125
Epoch 798/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.4455 - accuracy: 0.8125
Epoch 799/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.4469 - accuracy: 0.8125
Epoch 800/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.4499 - accuracy: 0.8125
Epoch 801/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.4588 - accuracy: 0.8125
Epoch 802/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.4497 - accuracy: 0.8125
Epoch 803/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4479 - accuracy: 0.8000
Epoch 804/1000
80/80 [==============================] - 0s 147us/sample - loss: 0.4507 - accuracy: 0.8125
Epoch 805/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4433 - accuracy: 0.8250
Epoch 806/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4476 - accuracy: 0.8375
Epoch 807/1000
80/80 [==============================] - 0s 154us/sample - loss: 0.4463 - accuracy: 0.8250
Epoch 808/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.4450 - accuracy: 0.8500
Epoch 809/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4445 - accuracy: 0.8375
Epoch 810/1000
80/80 [==============================] - 0s 231us/sample - loss: 0.4438 - accuracy: 0.8125
Epoch 811/1000
80/80 [==============================] - 0s 167us/sample - loss: 0.4419 - accuracy: 0.8125
Epoch 812/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.4458 - accuracy: 0.8250
Epoch 813/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.4455 - accuracy: 0.8250
Epoch 814/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.4504 - accuracy: 0.8125
Epoch 815/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.4448 - accuracy: 0.8250
Epoch 816/1000
80/80 [==============================] - 0s 108us/sample - loss: 0.4386 - accuracy: 0.8250
Epoch 817/1000
```

```
80/80 [==============================] - 0s 137us/sample - loss: 0.4388 - accuracy: 0.8375
Epoch 818/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.4462 - accuracy: 0.8500
Epoch 819/1000
80/80 [==============================] - 0s 152us/sample - loss: 0.4511 - accuracy: 0.8250
Epoch 820/1000
80/80 [==============================] - 0s 143us/sample - loss: 0.4382 - accuracy: 0.8375
Epoch 821/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4437 - accuracy: 0.8250
Epoch 822/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.4412 - accuracy: 0.7875
Epoch 823/1000
80/80 [==============================] - 0s 103us/sample - loss: 0.4405 - accuracy: 0.8250
Epoch 824/1000
80/80 [==============================] - 0s 98us/sample - loss: 0.4425 - accuracy: 0.8125
Epoch 825/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.4446 - accuracy: 0.8125
Epoch 826/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4398 - accuracy: 0.8375
Epoch 827/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4342 - accuracy: 0.8375
Epoch 828/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4409 - accuracy: 0.8375
Epoch 829/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.4488 - accuracy: 0.8250
Epoch 830/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.4418 - accuracy: 0.8250
Epoch 831/1000
80/80 [==============================] - 0s 100us/sample - loss: 0.4413 - accuracy: 0.8250
Epoch 832/1000
80/80 [==============================] - 0s 163us/sample - loss: 0.4396 - accuracy: 0.8375
Epoch 833/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.4363 - accuracy: 0.8500
Epoch 834/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.4374 - accuracy: 0.8500
Epoch 835/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.4461 - accuracy: 0.8125
Epoch 836/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.4376 - accuracy: 0.8125
Epoch 837/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.4367 - accuracy: 0.8375
Epoch 838/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.4379 - accuracy: 0.8625
```

```
Epoch 839/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4344 - accuracy: 0.8500
Epoch 840/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.4344 - accuracy: 0.8000
Epoch 841/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4396 - accuracy: 0.8000
Epoch 842/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.4383 - accuracy: 0.8625
Epoch 843/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4414 - accuracy: 0.8625
Epoch 844/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.4373 - accuracy: 0.8250
Epoch 845/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.4355 - accuracy: 0.8375
Epoch 846/1000
80/80 [==============================] - 0s 101us/sample - loss: 0.4412 - accuracy: 0.8500
Epoch 847/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.4309 - accuracy: 0.8375
Epoch 848/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.4417 - accuracy: 0.8375
Epoch 849/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.4485 - accuracy: 0.8375
Epoch 850/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.4348 - accuracy: 0.8250
Epoch 851/1000
80/80 [==============================] - 0s 107us/sample - loss: 0.4344 - accuracy: 0.8250
Epoch 852/1000
80/80 [==============================] - 0s 121us/sample - loss: 0.4361 - accuracy: 0.8375
Epoch 853/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.4356 - accuracy: 0.8250
Epoch 854/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.4289 - accuracy: 0.8250
Epoch 855/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.4328 - accuracy: 0.8125
Epoch 856/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.4380 - accuracy: 0.8375
Epoch 857/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.4374 - accuracy: 0.8375
Epoch 858/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4288 - accuracy: 0.8750
Epoch 859/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4328 - accuracy: 0.8500
Epoch 860/1000
```

```
80/80 [==============================] - 0s 143us/sample - loss: 0.4342 - accuracy: 0.8625
Epoch 861/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.4291 - accuracy: 0.8625
Epoch 862/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.4298 - accuracy: 0.8250
Epoch 863/1000
80/80 [==============================] - 0s 202us/sample - loss: 0.4330 - accuracy: 0.8250
Epoch 864/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4316 - accuracy: 0.8375
Epoch 865/1000
80/80 [==============================] - 0s 160us/sample - loss: 0.4325 - accuracy: 0.8125
Epoch 866/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.4323 - accuracy: 0.8125
Epoch 867/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4320 - accuracy: 0.8125
Epoch 868/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4279 - accuracy: 0.8250
Epoch 869/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.4304 - accuracy: 0.8125
Epoch 870/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.4295 - accuracy: 0.8250
Epoch 871/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.4307 - accuracy: 0.8500
Epoch 872/1000
80/80 [==============================] - 0s 112us/sample - loss: 0.4292 - accuracy: 0.8500
Epoch 873/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.4256 - accuracy: 0.8125
Epoch 874/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.4280 - accuracy: 0.8125
Epoch 875/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4287 - accuracy: 0.8125
Epoch 876/1000
80/80 [==============================] - 0s 127us/sample - loss: 0.4305 - accuracy: 0.8500
Epoch 877/1000
80/80 [==============================] - 0s 98us/sample - loss: 0.4287 - accuracy: 0.8500
Epoch 878/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.4309 - accuracy: 0.8125
Epoch 879/1000
80/80 [==============================] - 0s 107us/sample - loss: 0.4295 - accuracy: 0.8625
Epoch 880/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.4319 - accuracy: 0.8375
Epoch 881/1000
80/80 [==============================] - 0s 126us/sample - loss: 0.4298 - accuracy: 0.8375
```

```
Epoch 882/1000
80/80 [==============================] - 0s 147us/sample - loss: 0.4384 - accuracy: 0.8000
Epoch 883/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4302 - accuracy: 0.8250
Epoch 884/1000
80/80 [==============================] - 0s 97us/sample - loss: 0.4269 - accuracy: 0.8375
Epoch 885/1000
80/80 [==============================] - 0s 172us/sample - loss: 0.4279 - accuracy: 0.8625
Epoch 886/1000
80/80 [==============================] - 0s 104us/sample - loss: 0.4301 - accuracy: 0.8375
Epoch 887/1000
80/80 [==============================] - 0s 259us/sample - loss: 0.4310 - accuracy: 0.8375
Epoch 888/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.4249 - accuracy: 0.8250
Epoch 889/1000
80/80 [==============================] - 0s 162us/sample - loss: 0.4224 - accuracy: 0.8625
Epoch 890/1000
80/80 [==============================] - 0s 115us/sample - loss: 0.4218 - accuracy: 0.8375
Epoch 891/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.4198 - accuracy: 0.8500
Epoch 892/1000
80/80 [==============================] - 0s 157us/sample - loss: 0.4265 - accuracy: 0.8125
Epoch 893/1000
80/80 [==============================] - 0s 159us/sample - loss: 0.4302 - accuracy: 0.8125
Epoch 894/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.4195 - accuracy: 0.8375
Epoch 895/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.4239 - accuracy: 0.8375
Epoch 896/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.4313 - accuracy: 0.8125
Epoch 897/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.4260 - accuracy: 0.8500
Epoch 898/1000
80/80 [==============================] - 0s 110us/sample - loss: 0.4239 - accuracy: 0.8375
Epoch 899/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.4244 - accuracy: 0.8250
Epoch 900/1000
80/80 [==============================] - 0s 118us/sample - loss: 0.4229 - accuracy: 0.8250
Epoch 901/1000
80/80 [==============================] - 0s 152us/sample - loss: 0.4209 - accuracy: 0.8250
Epoch 902/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.4161 - accuracy: 0.8375
Epoch 903/1000
```

```
80/80 [==============================] - 0s 128us/sample - loss: 0.4234 - accuracy: 0.8500
Epoch 904/1000
80/80 [==============================] - 0s 150us/sample - loss: 0.4209 - accuracy: 0.8625
Epoch 905/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.4145 - accuracy: 0.8250
Epoch 906/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.4261 - accuracy: 0.8250
Epoch 907/1000
80/80 [==============================] - 0s 93us/sample - loss: 0.4273 - accuracy: 0.8375
Epoch 908/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.4194 - accuracy: 0.8625
Epoch 909/1000
80/80 [==============================] - 0s 122us/sample - loss: 0.4196 - accuracy: 0.8250
Epoch 910/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.4261 - accuracy: 0.8125
Epoch 911/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.4212 - accuracy: 0.8375
Epoch 912/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.4212 - accuracy: 0.8625
Epoch 913/1000
80/80 [==============================] - 0s 161us/sample - loss: 0.4143 - accuracy: 0.8500
Epoch 914/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.4167 - accuracy: 0.8625
Epoch 915/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4179 - accuracy: 0.8625
Epoch 916/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.4160 - accuracy: 0.8375
Epoch 917/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4177 - accuracy: 0.8250
Epoch 918/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.4145 - accuracy: 0.8250
Epoch 919/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.4159 - accuracy: 0.8500
Epoch 920/1000
80/80 [==============================] - 0s 175us/sample - loss: 0.4161 - accuracy: 0.8500
Epoch 921/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.4158 - accuracy: 0.8625
Epoch 922/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.4165 - accuracy: 0.8375
Epoch 923/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.4225 - accuracy: 0.8375
Epoch 924/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.4169 - accuracy: 0.8500
```

```
Epoch 925/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4145 - accuracy: 0.8750
Epoch 926/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.4125 - accuracy: 0.8625
Epoch 927/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4129 - accuracy: 0.8625
Epoch 928/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.4157 - accuracy: 0.8375
Epoch 929/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.4098 - accuracy: 0.8250
Epoch 930/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.4125 - accuracy: 0.8250
Epoch 931/1000
80/80 [==============================] - 0s 124us/sample - loss: 0.4143 - accuracy: 0.8000
Epoch 932/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.4204 - accuracy: 0.8250
Epoch 933/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.4126 - accuracy: 0.8750
Epoch 934/1000
80/80 [==============================] - 0s 117us/sample - loss: 0.4191 - accuracy: 0.8500
Epoch 935/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.4168 - accuracy: 0.8250
Epoch 936/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.4247 - accuracy: 0.8125
Epoch 937/1000
80/80 [==============================] - 0s 154us/sample - loss: 0.4163 - accuracy: 0.8125
Epoch 938/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.4133 - accuracy: 0.8625
Epoch 939/1000
80/80 [==============================] - 0s 146us/sample - loss: 0.4192 - accuracy: 0.8500
Epoch 940/1000
80/80 [==============================] - 0s 156us/sample - loss: 0.4088 - accuracy: 0.8500
Epoch 941/1000
80/80 [==============================] - 0s 142us/sample - loss: 0.4102 - accuracy: 0.8250
Epoch 942/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.4189 - accuracy: 0.8125
Epoch 943/1000
80/80 [==============================] - 0s 157us/sample - loss: 0.4181 - accuracy: 0.8125
Epoch 944/1000
80/80 [==============================] - 0s 154us/sample - loss: 0.4176 - accuracy: 0.8500
Epoch 945/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.4132 - accuracy: 0.8750
Epoch 946/1000
```

```
80/80 [==============================] - 0s 120us/sample - loss: 0.4072 - accuracy: 0.8500
Epoch 947/1000
80/80 [==============================] - 0s 140us/sample - loss: 0.4156 - accuracy: 0.8375
Epoch 948/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.4149 - accuracy: 0.8250
Epoch 949/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.4089 - accuracy: 0.8500
Epoch 950/1000
80/80 [==============================] - 0s 151us/sample - loss: 0.4096 - accuracy: 0.8500
Epoch 951/1000
80/80 [==============================] - 0s 152us/sample - loss: 0.4041 - accuracy: 0.8625
Epoch 952/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.4051 - accuracy: 0.8500
Epoch 953/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.4052 - accuracy: 0.8250
Epoch 954/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.4048 - accuracy: 0.8375
Epoch 955/1000
80/80 [==============================] - 0s 147us/sample - loss: 0.4067 - accuracy: 0.8500
Epoch 956/1000
80/80 [==============================] - 0s 148us/sample - loss: 0.4048 - accuracy: 0.8750
Epoch 957/1000
80/80 [==============================] - 0s 129us/sample - loss: 0.4079 - accuracy: 0.8500
Epoch 958/1000
80/80 [==============================] - 0s 112us/sample - loss: 0.4093 - accuracy: 0.8500
Epoch 959/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.4039 - accuracy: 0.8625
Epoch 960/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.4102 - accuracy: 0.8500
Epoch 961/1000
80/80 [==============================] - 0s 144us/sample - loss: 0.4112 - accuracy: 0.8625
Epoch 962/1000
80/80 [==============================] - 0s 268us/sample - loss: 0.4068 - accuracy: 0.8250
Epoch 963/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.4073 - accuracy: 0.8625
Epoch 964/1000
80/80 [==============================] - 0s 134us/sample - loss: 0.4131 - accuracy: 0.8500
Epoch 965/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.4087 - accuracy: 0.8500
Epoch 966/1000
80/80 [==============================] - 0s 110us/sample - loss: 0.4138 - accuracy: 0.8375
Epoch 967/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.4102 - accuracy: 0.8250
```

```
Epoch 968/1000
80/80 [==============================] - 0s 131us/sample - loss: 0.4073 - accuracy: 0.8500
Epoch 969/1000
80/80 [==============================] - 0s 145us/sample - loss: 0.4160 - accuracy: 0.8250
Epoch 970/1000
80/80 [==============================] - 0s 111us/sample - loss: 0.4027 - accuracy: 0.8250
Epoch 971/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.4090 - accuracy: 0.8250
Epoch 972/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.4061 - accuracy: 0.8250
Epoch 973/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.4072 - accuracy: 0.8625
Epoch 974/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.4097 - accuracy: 0.8375
Epoch 975/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.4028 - accuracy: 0.8625
Epoch 976/1000
80/80 [==============================] - 0s 113us/sample - loss: 0.4072 - accuracy: 0.8250
Epoch 977/1000
80/80 [==============================] - 0s 119us/sample - loss: 0.4075 - accuracy: 0.8250
Epoch 978/1000
80/80 [==============================] - 0s 109us/sample - loss: 0.4062 - accuracy: 0.8375
Epoch 979/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.4051 - accuracy: 0.8625
Epoch 980/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.4065 - accuracy: 0.8625
Epoch 981/1000
80/80 [==============================] - 0s 132us/sample - loss: 0.4080 - accuracy: 0.8375
Epoch 982/1000
80/80 [==============================] - 0s 147us/sample - loss: 0.4057 - accuracy: 0.8250
Epoch 983/1000
80/80 [==============================] - 0s 138us/sample - loss: 0.4050 - accuracy: 0.8375
Epoch 984/1000
80/80 [==============================] - 0s 233us/sample - loss: 0.4166 - accuracy: 0.8375
Epoch 985/1000
80/80 [==============================] - 0s 137us/sample - loss: 0.4029 - accuracy: 0.8500
Epoch 986/1000
80/80 [==============================] - 0s 133us/sample - loss: 0.4025 - accuracy: 0.8500
Epoch 987/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.4068 - accuracy: 0.7875
Epoch 988/1000
80/80 [==============================] - 0s 128us/sample - loss: 0.3994 - accuracy: 0.8375
Epoch 989/1000
```

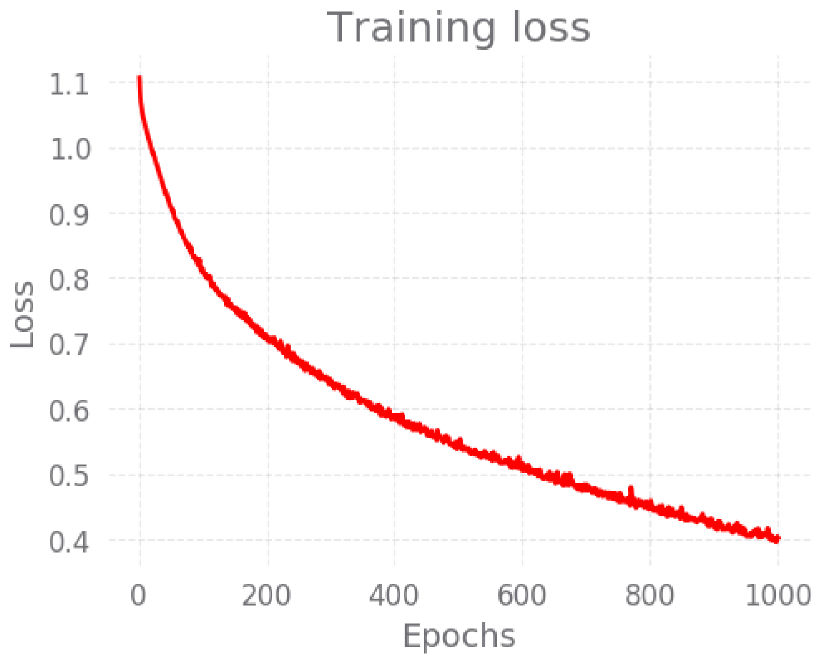
```
80/80 [==============================] - 0s 121us/sample - loss: 0.3990 - accuracy: 0.8500
Epoch 990/1000
80/80 [==============================] - 0s 130us/sample - loss: 0.4048 - accuracy: 0.8250
Epoch 991/1000
80/80 [==============================] - 0s 149us/sample - loss: 0.4021 - accuracy: 0.8500
Epoch 992/1000
80/80 [==============================] - 0s 99us/sample - loss: 0.3987 - accuracy: 0.8375
Epoch 993/1000
80/80 [==============================] - 0s 136us/sample - loss: 0.4022 - accuracy: 0.8250
Epoch 994/1000
80/80 [==============================] - 0s 135us/sample - loss: 0.3983 - accuracy: 0.8250
Epoch 995/1000
80/80 [==============================] - 0s 139us/sample - loss: 0.3980 - accuracy: 0.8500
Epoch 996/1000
80/80 [==============================] - 0s 125us/sample - loss: 0.3990 - accuracy: 0.8625
Epoch 997/1000
80/80 [==============================] - 0s 123us/sample - loss: 0.3954 - accuracy: 0.8500
Epoch 998/1000
80/80 [==============================] - 0s 120us/sample - loss: 0.3962 - accuracy: 0.8625
Epoch 999/1000
80/80 [==============================] - 0s 114us/sample - loss: 0.4038 - accuracy: 0.8250
Epoch 1000/1000
80/80 [==============================] - 0s 116us/sample - loss: 0.4019 - accuracy: 0.8250

history_list ['loss', 'accuracy']
```

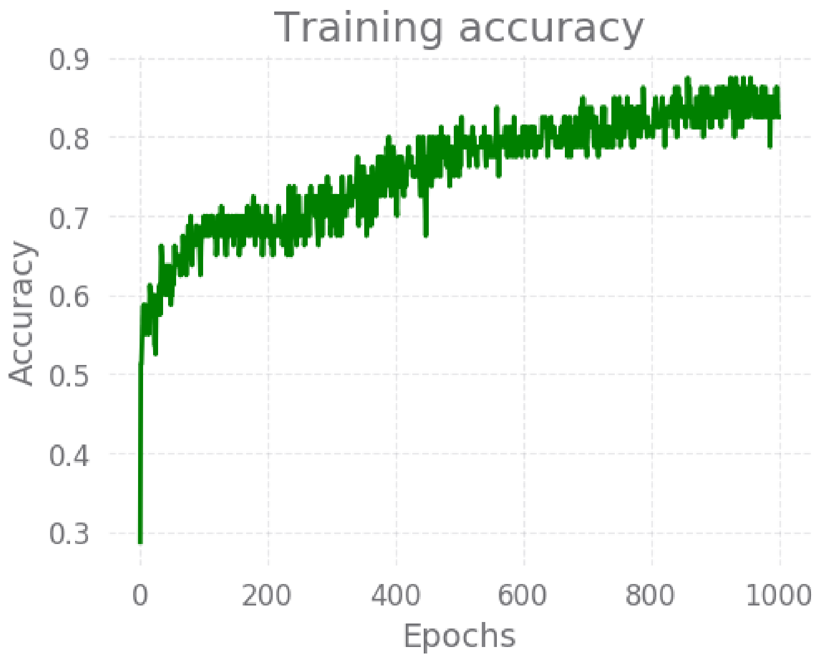
In [0]:

```
#@title ## Evaluating

evaluation = overfit_model.evaluate(ov_x_test, ov_y_test_c, verbose=0)

print('Loss: %.3f\t Accuracy: %.2f%%' % (evaluation[0], evaluation[1] * 100))
```

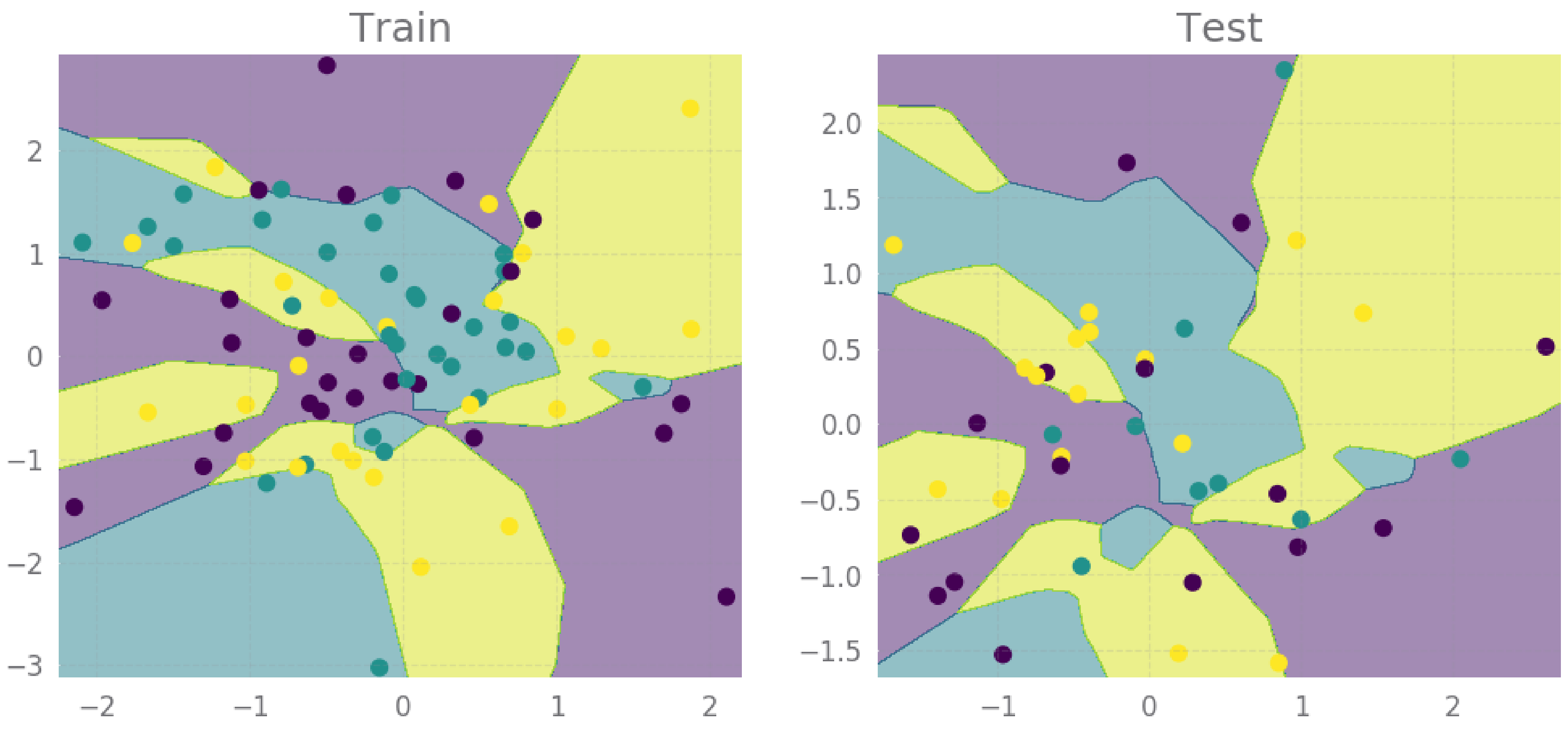
```
Loss: 1.959      Accuracy: 52.50%
```

In [0]:
```
#@title ## Plotting

xfig = 12.0 #@param {type:"number"}
yfig = 4.0 #@param {type:"number"}

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(xfig, yfig))


loss = history_dict[history_list[history_list.index('loss')]]
acc = history_dict[history_list[history_list.index('accuracy')]]

epochs = range(1, len(loss) + 1)


ax1.plot(epochs, loss, 'r', label='Training loss')
ax1.set_title('Training loss')
ax1.set_xlabel('Epochs')
ax1.set_ylabel('Loss')

ax2.plot(epochs, acc, 'g', label='Training accuracy')
ax2.set_title('Training accuracy')
ax2.set_xlabel('Epochs')
ax2.set_ylabel('Accuracy')

plt.show()
```


Training loss
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
Loss
0
200
400
600
800
1000
Epochs

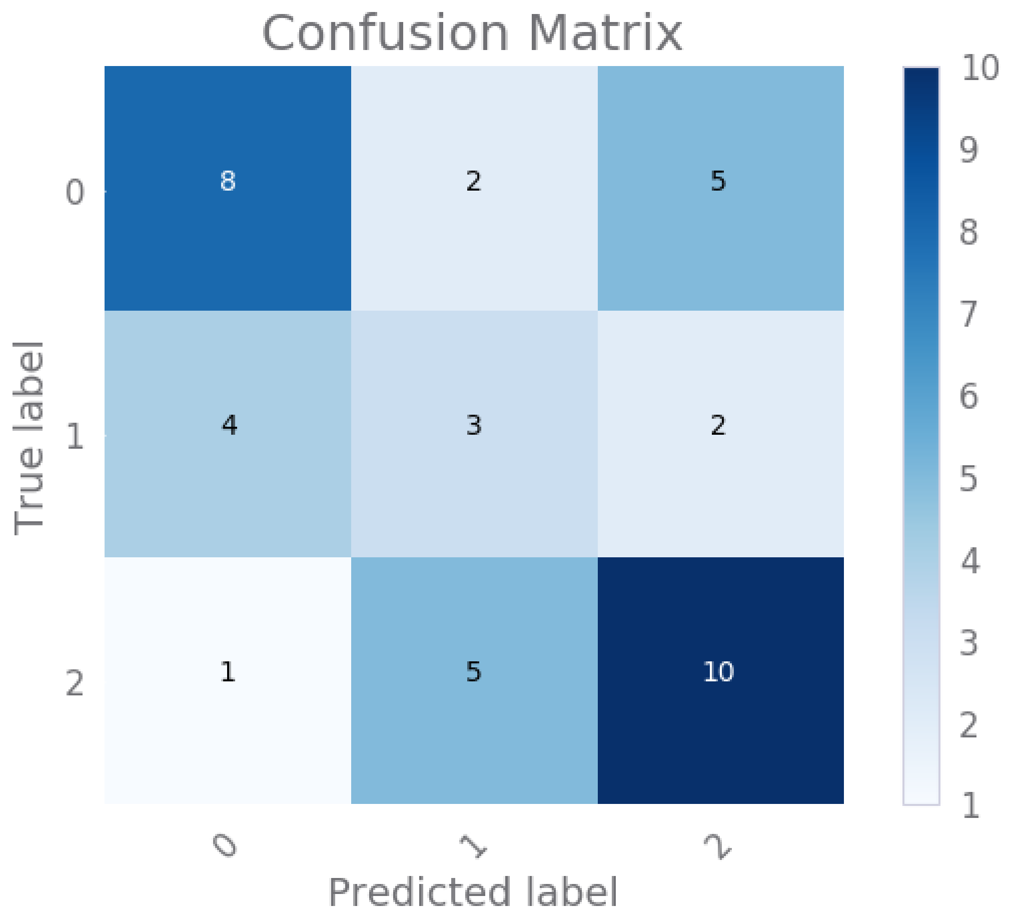

Training accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Accuracy
0
200
400
600
800
1000
Epochs

In [0]:
```
#@title ## Visualize the decision boundary

xfig = 12.0 #@param {type:"number"}
yfig = 5.0 #@param {type:"number"}

plt.figure(figsize=(xfig, yfig))
plt.subplot(1, 2, 1)
plt.title('Train')
plot_mc_decision_boundary(model=overfit_model, x=ov_x_train, y=ov_y_train)
plt.subplot(1, 2, 2)
plt.title('Test')
plot_mc_decision_boundary(model=overfit_model, x=ov_x_test, y=ov_y_test)
plt.show()
```



In [0]:
```
#@title ## Confusion matrix

pred_test = overfit_model.predict(ov_x_test)
pred_test = np.argmax(pred_test, axis=1)

cm = confusion_matrix(ov_y_test, pred_test)
plot_confusion_matrix(cm=cm, classes=[0, 1, 2])

print(classification_report(ov_y_test, pred_test))
```

```
              precision    recall  f1-score   support

           0       0.62      0.53      0.57        15
           1       0.30      0.33      0.32         9
           2       0.59      0.62      0.61        16

    accuracy                           0.53        40
   macro avg       0.50      0.50      0.50        40
weighted avg       0.53      0.53      0.53        40
```



Regularization

In [0]:
```
#@title ## Build non-linear model [MLP Model](https://en.wikipedia.org/wiki/Multilayer_perceptron)

def build_non_linear_with_regularization(n_classes, n_units, n_features, compiling=True, opt='adam',
loss='categorical_crossentropy', metrics=None, reg_rate=0.001):
    """ Build a linear model [MLP Model].

    Args:
        n_classes (int): dimensionality of the output space.
        n_units (int): dimensionality of the hidden layer.
        n_features (int): indicates that the expected input will be batches of n_features-dimensional
vectors.
        compiling (boolean): whether the model complied or not.
        opt (str, keras.optimizers): optimizer instance.
        loss (str, keras.losses): objective function.
        metrics (str, keras.metrics): List of metrics to be evaluated by the model during training an
d testing.
        reg_rate (float): regulaization rate.

    Returns:
        model (keras.Model): a keras model.
    """

    l1 = tf.keras.regularizers.l1
    l1 = l1(reg_rate)

    model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(n_features,)),
        tf.keras.layers.Dense(n_units, activation='relu', activity_regularizer=l1),
        tf.keras.layers.Dense(n_classes, activation='softmax')
    ])

    if compiling:

        if metrics is None or not isinstance(metrics, list):
            metrics = ['accuracy']

        model.compile(optimizer=opt, loss=loss, metrics=metrics)

    return model
```

In [0]:
```
#@title ## Create non-linear model

compiling = True #@param {type:"boolean"}

n_units = 5 #@param {type:"integer"}
opt = 'adam' #@param {type:"string"}
loss = 'categorical_crossentropy' #@param {type:"string"}
metrics = "accuracy" #@param ["accuracy"] {allow-input: true}
reg_rate = 0.001 #@param {type:"number"}

n_classes = y_train_c.shape[1] # 3, num_classes
n_features = x_train.shape[1] # 2, dimensions


nl_model_with_reg = build_non_linear_with_regularization(n_classes, n_units, n_features, compiling, o
pt, loss, metrics, reg_rate)
nl_model_with_reg
```

Out[0]:
```
<tensorflow.python.keras.engine.sequential.Sequential at 0x7fa4530a8278>
```

In [0]:
```
#@title ## Fit non-linear model with regularization.

r = nl_model_with_reg.fit(x_train, y_train_c, validation_split=0.1, epochs=100, verbose=1)

history_dict = r.history
history_list = list(history_dict.keys())

print()
print('history_list', history_list)
```

```
Train on 904 samples, validate on 101 samples
Epoch 1/100
904/904 [==============================] - 0s 195us/sample - loss: 0.6156 - accuracy: 0.6084 - val_l
oss: 0.6238 - val_accuracy: 0.6337
Epoch 2/100
904/904 [==============================] - 0s 89us/sample - loss: 0.6131 - accuracy: 0.6073 - val_lo
ss: 0.6221 - val_accuracy: 0.6337
Epoch 3/100
904/904 [==============================] - 0s 95us/sample - loss: 0.6104 - accuracy: 0.6139 - val_lo
ss: 0.6189 - val_accuracy: 0.6337
Epoch 4/100
904/904 [==============================] - 0s 94us/sample - loss: 0.6080 - accuracy: 0.6150 - val_lo
ss: 0.6176 - val_accuracy: 0.6337
Epoch 5/100
904/904 [==============================] - 0s 95us/sample - loss: 0.6056 - accuracy: 0.6261 - val_lo
ss: 0.6146 - val_accuracy: 0.6337
Epoch 6/100
904/904 [==============================] - 0s 96us/sample - loss: 0.6033 - accuracy: 0.6228 - val_lo
ss: 0.6132 - val_accuracy: 0.6436
Epoch 7/100
904/904 [==============================] - 0s 90us/sample - loss: 0.6013 - accuracy: 0.6294 - val_lo
ss: 0.6110 - val_accuracy: 0.6436
Epoch 8/100
904/904 [==============================] - 0s 88us/sample - loss: 0.5987 - accuracy: 0.6316 - val_lo
ss: 0.6063 - val_accuracy: 0.6436
Epoch 9/100
904/904 [==============================] - 0s 101us/sample - loss: 0.5965 - accuracy: 0.6394 - val_l
oss: 0.6057 - val_accuracy: 0.6436
Epoch 10/100
904/904 [==============================] - 0s 97us/sample - loss: 0.5939 - accuracy: 0.6449 - val_lo
ss: 0.6030 - val_accuracy: 0.6436
Epoch 11/100
904/904 [==============================] - 0s 90us/sample - loss: 0.5916 - accuracy: 0.6438 - val_lo
ss: 0.6001 - val_accuracy: 0.6436
Epoch 12/100
904/904 [==============================] - 0s 98us/sample - loss: 0.5892 - accuracy: 0.6394 - val_lo
ss: 0.5973 - val_accuracy: 0.6436
Epoch 13/100
904/904 [==============================] - 0s 88us/sample - loss: 0.5868 - accuracy: 0.6549 - val_lo
ss: 0.5956 - val_accuracy: 0.6436
Epoch 14/100
904/904 [==============================] - 0s 93us/sample - loss: 0.5844 - accuracy: 0.6482 - val_lo
ss: 0.5916 - val_accuracy: 0.6436
```

```
Epoch 15/100
904/904 [==============================] - 0s 104us/sample - loss: 0.5822 - accuracy: 0.6571 - val_l
oss: 0.5882 - val_accuracy: 0.6436
Epoch 16/100
904/904 [==============================] - 0s 94us/sample - loss: 0.5795 - accuracy: 0.6560 - val_lo
ss: 0.5876 - val_accuracy: 0.6535
Epoch 17/100
904/904 [==============================] - 0s 94us/sample - loss: 0.5769 - accuracy: 0.6571 - val_lo
ss: 0.5846 - val_accuracy: 0.6535
Epoch 18/100
904/904 [==============================] - 0s 92us/sample - loss: 0.5749 - accuracy: 0.6549 - val_lo
ss: 0.5818 - val_accuracy: 0.6535
Epoch 19/100
904/904 [==============================] - 0s 91us/sample - loss: 0.5721 - accuracy: 0.6759 - val_lo
ss: 0.5795 - val_accuracy: 0.6634
Epoch 20/100
904/904 [==============================] - 0s 96us/sample - loss: 0.5698 - accuracy: 0.6803 - val_lo
ss: 0.5774 - val_accuracy: 0.6634
Epoch 21/100
904/904 [==============================] - 0s 94us/sample - loss: 0.5671 - accuracy: 0.6814 - val_lo
ss: 0.5745 - val_accuracy: 0.6733
Epoch 22/100
904/904 [==============================] - 0s 94us/sample - loss: 0.5648 - accuracy: 0.6847 - val_lo
ss: 0.5717 - val_accuracy: 0.6634
Epoch 23/100
904/904 [==============================] - 0s 91us/sample - loss: 0.5623 - accuracy: 0.6847 - val_lo
ss: 0.5690 - val_accuracy: 0.6832
Epoch 24/100
904/904 [==============================] - 0s 91us/sample - loss: 0.5603 - accuracy: 0.6803 - val_lo
ss: 0.5655 - val_accuracy: 0.6733
Epoch 25/100
904/904 [==============================] - 0s 88us/sample - loss: 0.5572 - accuracy: 0.6847 - val_lo
ss: 0.5637 - val_accuracy: 0.6832
Epoch 26/100
904/904 [==============================] - 0s 79us/sample - loss: 0.5551 - accuracy: 0.6958 - val_lo
ss: 0.5607 - val_accuracy: 0.6931
Epoch 27/100
904/904 [==============================] - 0s 87us/sample - loss: 0.5522 - accuracy: 0.6969 - val_lo
ss: 0.5586 - val_accuracy: 0.6931
Epoch 28/100
904/904 [==============================] - 0s 76us/sample - loss: 0.5497 - accuracy: 0.6991 - val_lo
ss: 0.5562 - val_accuracy: 0.6931
Epoch 29/100
```

```
904/904 [==============================] - 0s 82us/sample - loss: 0.5476 - accuracy: 0.7080 - val_lo
ss: 0.5549 - val_accuracy: 0.6931
Epoch 30/100
904/904 [==============================] - 0s 80us/sample - loss: 0.5447 - accuracy: 0.7135 - val_lo
ss: 0.5513 - val_accuracy: 0.6931
Epoch 31/100
904/904 [==============================] - 0s 77us/sample - loss: 0.5423 - accuracy: 0.7102 - val_lo
ss: 0.5476 - val_accuracy: 0.7030
Epoch 32/100
904/904 [==============================] - 0s 75us/sample - loss: 0.5397 - accuracy: 0.7168 - val_lo
ss: 0.5463 - val_accuracy: 0.7030
Epoch 33/100
904/904 [==============================] - 0s 77us/sample - loss: 0.5374 - accuracy: 0.7257 - val_lo
ss: 0.5449 - val_accuracy: 0.6931
Epoch 34/100
904/904 [==============================] - 0s 80us/sample - loss: 0.5348 - accuracy: 0.7223 - val_lo
ss: 0.5403 - val_accuracy: 0.7030
Epoch 35/100
904/904 [==============================] - 0s 77us/sample - loss: 0.5322 - accuracy: 0.7212 - val_lo
ss: 0.5365 - val_accuracy: 0.7030
Epoch 36/100
904/904 [==============================] - 0s 76us/sample - loss: 0.5296 - accuracy: 0.7246 - val_lo
ss: 0.5346 - val_accuracy: 0.7030
Epoch 37/100
904/904 [==============================] - 0s 76us/sample - loss: 0.5268 - accuracy: 0.7312 - val_lo
ss: 0.5322 - val_accuracy: 0.7030
Epoch 38/100
904/904 [==============================] - 0s 80us/sample - loss: 0.5243 - accuracy: 0.7356 - val_lo
ss: 0.5298 - val_accuracy: 0.7030
Epoch 39/100
904/904 [==============================] - 0s 77us/sample - loss: 0.5217 - accuracy: 0.7334 - val_lo
ss: 0.5292 - val_accuracy: 0.7030
Epoch 40/100
904/904 [==============================] - 0s 79us/sample - loss: 0.5190 - accuracy: 0.7400 - val_lo
ss: 0.5268 - val_accuracy: 0.7030
Epoch 41/100
904/904 [==============================] - 0s 82us/sample - loss: 0.5167 - accuracy: 0.7400 - val_lo
ss: 0.5238 - val_accuracy: 0.7030
Epoch 42/100
904/904 [==============================] - 0s 82us/sample - loss: 0.5144 - accuracy: 0.7412 - val_lo
ss: 0.5205 - val_accuracy: 0.7030
Epoch 43/100
904/904 [==============================] - 0s 78us/sample - loss: 0.5115 - accuracy: 0.7412 - val_lo
```

```
ss: 0.5204 - val_accuracy: 0.7030
Epoch 44/100
904/904 [==============================] - 0s 75us/sample - loss: 0.5091 - accuracy: 0.7456 - val_lo
ss: 0.5172 - val_accuracy: 0.7030
Epoch 45/100
904/904 [==============================] - 0s 79us/sample - loss: 0.5067 - accuracy: 0.7423 - val_lo
ss: 0.5140 - val_accuracy: 0.7030
Epoch 46/100
904/904 [==============================] - 0s 76us/sample - loss: 0.5041 - accuracy: 0.7500 - val_lo
ss: 0.5121 - val_accuracy: 0.7030
Epoch 47/100
904/904 [==============================] - 0s 82us/sample - loss: 0.5015 - accuracy: 0.7533 - val_lo
ss: 0.5085 - val_accuracy: 0.7228
Epoch 48/100
904/904 [==============================] - 0s 80us/sample - loss: 0.4989 - accuracy: 0.7511 - val_lo
ss: 0.5051 - val_accuracy: 0.7228
Epoch 49/100
904/904 [==============================] - 0s 78us/sample - loss: 0.4967 - accuracy: 0.7544 - val_lo
ss: 0.5035 - val_accuracy: 0.7228
Epoch 50/100
904/904 [==============================] - 0s 78us/sample - loss: 0.4939 - accuracy: 0.7577 - val_lo
ss: 0.5015 - val_accuracy: 0.7327
Epoch 51/100
904/904 [==============================] - 0s 83us/sample - loss: 0.4918 - accuracy: 0.7566 - val_lo
ss: 0.4994 - val_accuracy: 0.7426
Epoch 52/100
904/904 [==============================] - 0s 79us/sample - loss: 0.4895 - accuracy: 0.7655 - val_lo
ss: 0.4983 - val_accuracy: 0.7327
Epoch 53/100
904/904 [==============================] - 0s 78us/sample - loss: 0.4877 - accuracy: 0.7688 - val_lo
ss: 0.4946 - val_accuracy: 0.7327
Epoch 54/100
904/904 [==============================] - 0s 90us/sample - loss: 0.4848 - accuracy: 0.7699 - val_lo
ss: 0.4929 - val_accuracy: 0.7426
Epoch 55/100
904/904 [==============================] - 0s 79us/sample - loss: 0.4826 - accuracy: 0.7655 - val_lo
ss: 0.4893 - val_accuracy: 0.7426
Epoch 56/100
904/904 [==============================] - 0s 76us/sample - loss: 0.4804 - accuracy: 0.7699 - val_lo
ss: 0.4868 - val_accuracy: 0.7426
Epoch 57/100
904/904 [==============================] - 0s 77us/sample - loss: 0.4782 - accuracy: 0.7688 - val_lo
ss: 0.4833 - val_accuracy: 0.7525
```

```
Epoch 58/100
904/904 [==============================] - 0s 79us/sample - loss: 0.4757 - accuracy: 0.7688 - val_lo
ss: 0.4803 - val_accuracy: 0.7624
Epoch 59/100
904/904 [==============================] - 0s 77us/sample - loss: 0.4735 - accuracy: 0.7677 - val_lo
ss: 0.4794 - val_accuracy: 0.7624
Epoch 60/100
904/904 [==============================] - 0s 77us/sample - loss: 0.4713 - accuracy: 0.7743 - val_lo
ss: 0.4767 - val_accuracy: 0.7624
Epoch 61/100
904/904 [==============================] - 0s 77us/sample - loss: 0.4689 - accuracy: 0.7765 - val_lo
ss: 0.4747 - val_accuracy: 0.7723
Epoch 62/100
904/904 [==============================] - 0s 79us/sample - loss: 0.4667 - accuracy: 0.7777 - val_lo
ss: 0.4725 - val_accuracy: 0.7723
Epoch 63/100
904/904 [==============================] - 0s 83us/sample - loss: 0.4646 - accuracy: 0.7799 - val_lo
ss: 0.4695 - val_accuracy: 0.7822
Epoch 64/100
904/904 [==============================] - 0s 79us/sample - loss: 0.4627 - accuracy: 0.7810 - val_lo
ss: 0.4665 - val_accuracy: 0.7921
Epoch 65/100
904/904 [==============================] - 0s 78us/sample - loss: 0.4603 - accuracy: 0.7799 - val_lo
ss: 0.4654 - val_accuracy: 0.7822
Epoch 66/100
904/904 [==============================] - 0s 77us/sample - loss: 0.4585 - accuracy: 0.7821 - val_lo
ss: 0.4636 - val_accuracy: 0.7822
Epoch 67/100
904/904 [==============================] - 0s 78us/sample - loss: 0.4564 - accuracy: 0.7821 - val_lo
ss: 0.4605 - val_accuracy: 0.7921
Epoch 68/100
904/904 [==============================] - 0s 86us/sample - loss: 0.4543 - accuracy: 0.7854 - val_lo
ss: 0.4602 - val_accuracy: 0.7822
Epoch 69/100
904/904 [==============================] - 0s 75us/sample - loss: 0.4523 - accuracy: 0.7854 - val_lo
ss: 0.4571 - val_accuracy: 0.7921
Epoch 70/100
904/904 [==============================] - 0s 76us/sample - loss: 0.4504 - accuracy: 0.7865 - val_lo
ss: 0.4544 - val_accuracy: 0.7921
Epoch 71/100
904/904 [==============================] - 0s 77us/sample - loss: 0.4484 - accuracy: 0.7854 - val_lo
ss: 0.4531 - val_accuracy: 0.7921
Epoch 72/100
```

```
904/904 [==============================] - 0s 77us/sample - loss: 0.4467 - accuracy: 0.7909 - val_lo
ss: 0.4522 - val_accuracy: 0.8020
Epoch 73/100
904/904 [==============================] - 0s 80us/sample - loss: 0.4444 - accuracy: 0.7920 - val_lo
ss: 0.4485 - val_accuracy: 0.8020
Epoch 74/100
904/904 [==============================] - 0s 75us/sample - loss: 0.4425 - accuracy: 0.7887 - val_lo
ss: 0.4457 - val_accuracy: 0.8020
Epoch 75/100
904/904 [==============================] - 0s 79us/sample - loss: 0.4409 - accuracy: 0.7920 - val_lo
ss: 0.4460 - val_accuracy: 0.7921
Epoch 76/100
904/904 [==============================] - 0s 75us/sample - loss: 0.4387 - accuracy: 0.7920 - val_lo
ss: 0.4432 - val_accuracy: 0.8020
Epoch 77/100
904/904 [==============================] - 0s 77us/sample - loss: 0.4367 - accuracy: 0.7931 - val_lo
ss: 0.4401 - val_accuracy: 0.8020
Epoch 78/100
904/904 [==============================] - 0s 78us/sample - loss: 0.4350 - accuracy: 0.7965 - val_lo
ss: 0.4377 - val_accuracy: 0.8119
Epoch 79/100
904/904 [==============================] - 0s 83us/sample - loss: 0.4331 - accuracy: 0.7987 - val_lo
ss: 0.4364 - val_accuracy: 0.8119
Epoch 80/100
904/904 [==============================] - 0s 78us/sample - loss: 0.4314 - accuracy: 0.7976 - val_lo
ss: 0.4342 - val_accuracy: 0.8119
Epoch 81/100
904/904 [==============================] - 0s 84us/sample - loss: 0.4296 - accuracy: 0.7987 - val_lo
ss: 0.4319 - val_accuracy: 0.8218
Epoch 82/100
904/904 [==============================] - 0s 96us/sample - loss: 0.4279 - accuracy: 0.8009 - val_lo
ss: 0.4304 - val_accuracy: 0.8218
Epoch 83/100
904/904 [==============================] - 0s 76us/sample - loss: 0.4261 - accuracy: 0.8042 - val_lo
ss: 0.4294 - val_accuracy: 0.8218
Epoch 84/100
904/904 [==============================] - 0s 79us/sample - loss: 0.4244 - accuracy: 0.8064 - val_lo
ss: 0.4274 - val_accuracy: 0.8218
Epoch 85/100
904/904 [==============================] - 0s 87us/sample - loss: 0.4229 - accuracy: 0.8086 - val_lo
ss: 0.4262 - val_accuracy: 0.8218
Epoch 86/100
904/904 [==============================] - 0s 76us/sample - loss: 0.4211 - accuracy: 0.8064 - val_lo
```

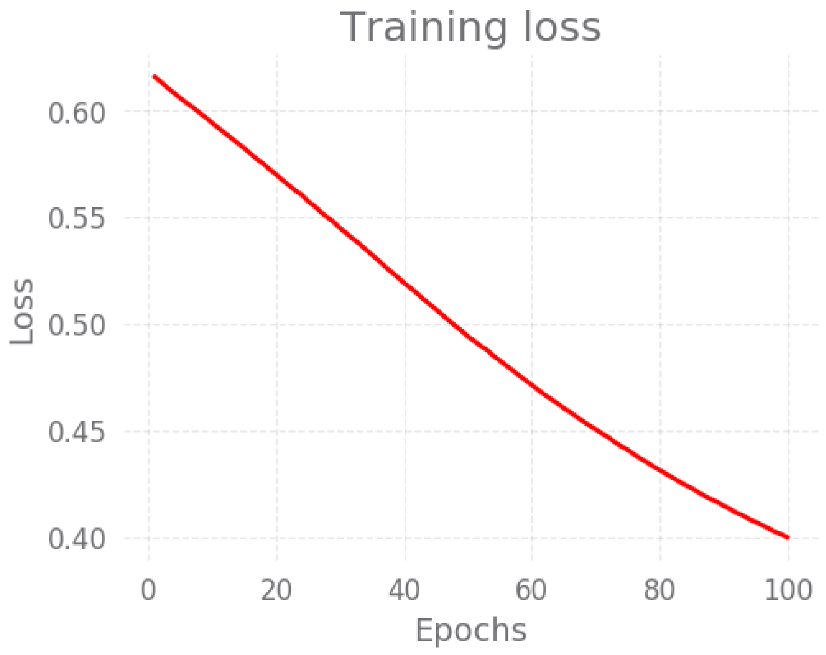
```
ss: 0.4234 - val_accuracy: 0.8218
Epoch 87/100
904/904 [==============================] - 0s 79us/sample - loss: 0.4194 - accuracy: 0.8064 - val_lo
ss: 0.4228 - val_accuracy: 0.8218
Epoch 88/100
904/904 [==============================] - 0s 80us/sample - loss: 0.4178 - accuracy: 0.8108 - val_lo
ss: 0.4213 - val_accuracy: 0.8317
Epoch 89/100
904/904 [==============================] - 0s 77us/sample - loss: 0.4165 - accuracy: 0.8131 - val_lo
ss: 0.4208 - val_accuracy: 0.8317
Epoch 90/100
904/904 [==============================] - 0s 76us/sample - loss: 0.4146 - accuracy: 0.8142 - val_lo
ss: 0.4174 - val_accuracy: 0.8317
Epoch 91/100
904/904 [==============================] - 0s 77us/sample - loss: 0.4132 - accuracy: 0.8142 - val_lo
ss: 0.4161 - val_accuracy: 0.8416
Epoch 92/100
904/904 [==============================] - 0s 77us/sample - loss: 0.4114 - accuracy: 0.8164 - val_lo
ss: 0.4154 - val_accuracy: 0.8317
Epoch 93/100
904/904 [==============================] - 0s 77us/sample - loss: 0.4102 - accuracy: 0.8153 - val_lo
ss: 0.4128 - val_accuracy: 0.8416
Epoch 94/100
904/904 [==============================] - 0s 79us/sample - loss: 0.4084 - accuracy: 0.8208 - val_lo
ss: 0.4124 - val_accuracy: 0.8416
Epoch 95/100
904/904 [==============================] - 0s 93us/sample - loss: 0.4071 - accuracy: 0.8208 - val_lo
ss: 0.4104 - val_accuracy: 0.8416
Epoch 96/100
904/904 [==============================] - 0s 81us/sample - loss: 0.4056 - accuracy: 0.8197 - val_lo
ss: 0.4083 - val_accuracy: 0.8515
Epoch 97/100
904/904 [==============================] - 0s 78us/sample - loss: 0.4041 - accuracy: 0.8208 - val_lo
ss: 0.4062 - val_accuracy: 0.8416
Epoch 98/100
904/904 [==============================] - 0s 81us/sample - loss: 0.4025 - accuracy: 0.8252 - val_lo
ss: 0.4049 - val_accuracy: 0.8515
Epoch 99/100
904/904 [==============================] - 0s 80us/sample - loss: 0.4013 - accuracy: 0.8219 - val_lo
ss: 0.4028 - val_accuracy: 0.8515
Epoch 100/100
904/904 [==============================] - 0s 83us/sample - loss: 0.3999 - accuracy: 0.8263 - val_lo
ss: 0.4023 - val_accuracy: 0.8515
```

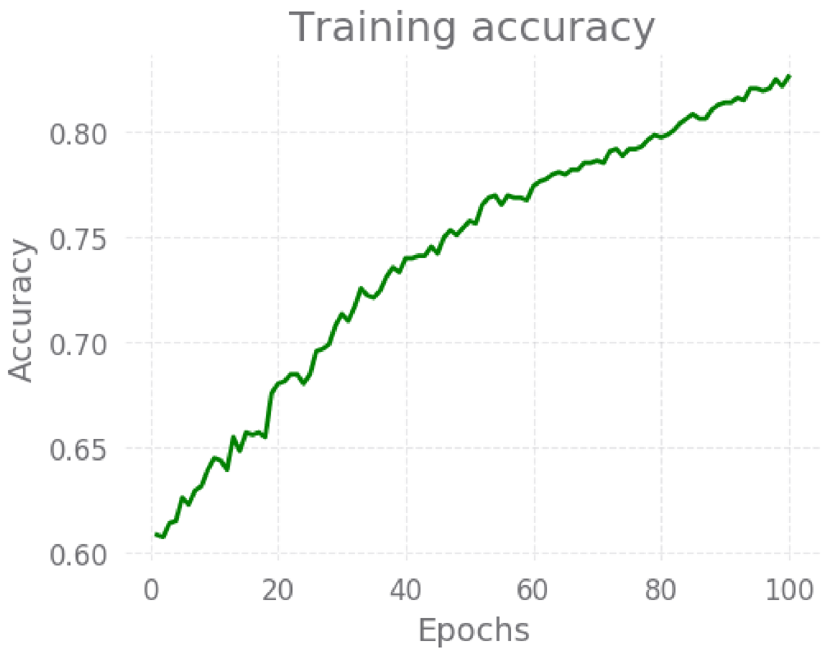
```
history_list ['loss', 'accuracy', 'val_loss', 'val_accuracy']
```

In [0]:
```
#@title ## Evaluating

evaluation = nl_model_with_reg.evaluate(x_test, y_test_c, verbose=0)

print('Loss: %.3f\t Accuracy: %.2f%%' % (evaluation[0], evaluation[1] * 100))
```

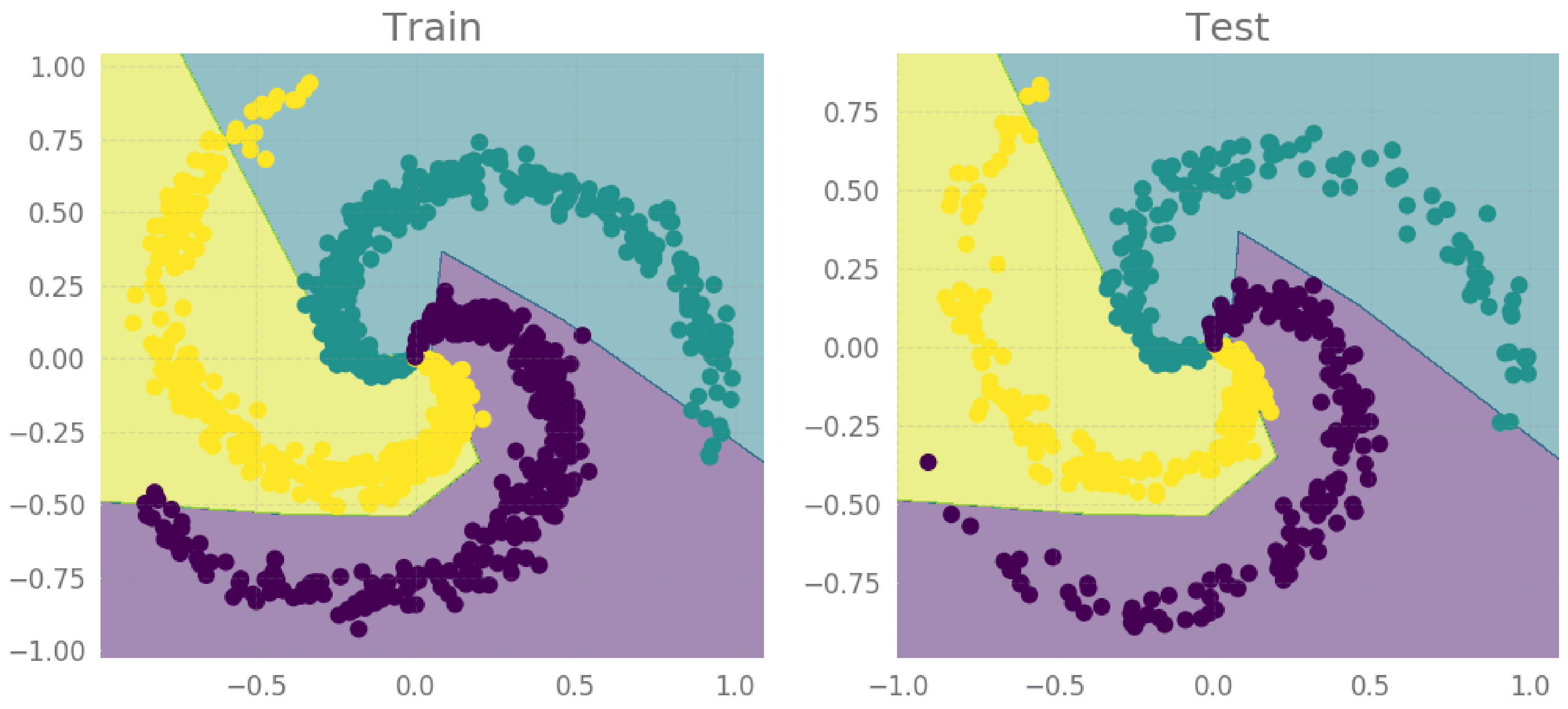
```
Loss: 0.392      Accuracy: 84.85%
```

In [0]:
```
#@title ## Plotting

xfig = 12.0 #@param {type:"number"}
yfig = 4.0 #@param {type:"number"}

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(xfig, yfig))


loss = history_dict[history_list[history_list.index('loss')]]
acc = history_dict[history_list[history_list.index('accuracy')]]

epochs = range(1, len(loss) + 1)


ax1.plot(epochs, loss, 'r', label='Training loss')
ax1.set_title('Training loss')
ax1.set_xlabel('Epochs')
ax1.set_ylabel('Loss')

ax2.plot(epochs, acc, 'g', label='Training accuracy')
ax2.set_title('Training accuracy')
ax2.set_xlabel('Epochs')
ax2.set_ylabel('Accuracy')

plt.show()
```


Training loss
Loss
0.60
0.55
0.50
0.45
0.40
0
20
40
60
80
100
Epochs

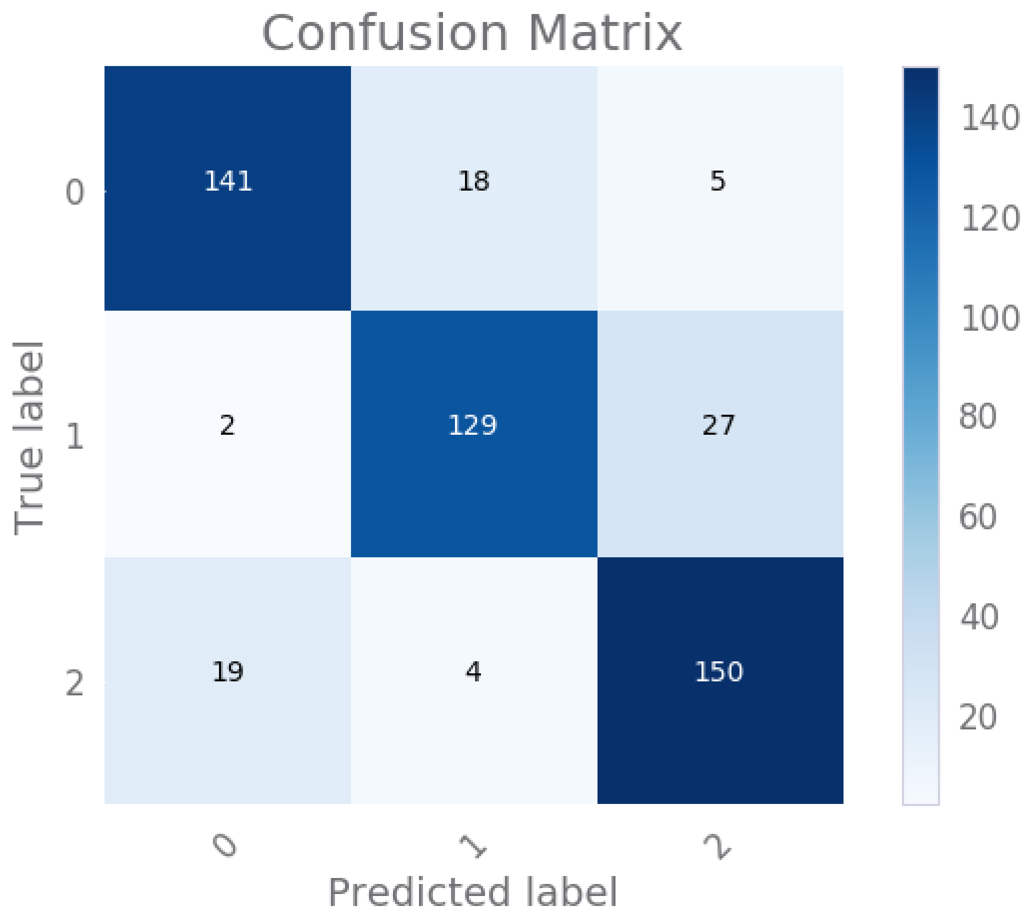

Training accuracy
Accuracy
0.80
0.75
0.70
0.65
0.60
0
20
40
60
80
100
Epochs

In [0]:
```
#@title ## Visualize the decision boundary

xfig = 12.0 #@param {type:"number"}
yfig = 5.0 #@param {type:"number"}

plt.figure(figsize=(xfig, yfig))
plt.subplot(1, 2, 1)
plt.title('Train')
plot_mc_decision_boundary(model=nl_model_with_reg, x=x_train, y=y_train)
plt.subplot(1, 2, 2)
plt.title('Test')
plot_mc_decision_boundary(model=nl_model_with_reg, x=x_test, y=y_test)
plt.show()
```



In [0]:
```
#@title ## Confusion matrix

pred_test = nl_model_with_reg.predict(x_test)
pred_test = np.argmax(pred_test, axis=1)

cm = confusion_matrix(y_test, pred_test)
plot_confusion_matrix(cm=cm, classes=[0, 1, 2])

print(classification_report(y_test, pred_test))
```

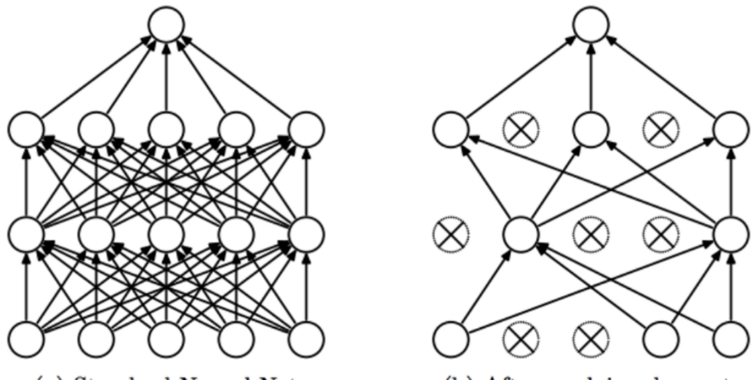
```
              precision    recall  f1-score   support

           0       0.87      0.86      0.87       164
           1       0.85      0.82      0.83       158
           2       0.82      0.87      0.85       173

    accuracy                           0.85       495
   macro avg       0.85      0.85      0.85       495
weighted avg       0.85      0.85      0.85       495
```



Dropout

A great technique to overcome overfitting is to increase the size of your data but this isn't always an option. Fortuntely, there are methods like regularization and dropout that can help create a more robust model. We've already seen regularization and we can easily add it in our optimizer to use it in PyTorch.

Dropout is a technique (used only during training) that allows us to zero the outputs of neurons. We do this for p% of the total neurons in each layer and it changes every batch. Dropout prevents units from co-adapting too much to the data and acts as a sampling strategy since we drop a different set of neurons each time.



In [0]:

```
#@title ## Build non-linear model [MLP Model](https://en.wikipedia.org/wiki/Multilayer_perceptron)

def build_non_linear_with_dropout(n_classes, n_units, n_features, compiling=True, opt='adam', loss='c
ategorical_crossentropy', metrics=None, drop_rate=0.5):
    """ Build a linear model [MLP Model].

    Args:
        n_classes (int): dimensionality of the output space.
        n_units (int): dimensionality of the hidden layer.
        n_features (int): indicates that the expected input will be batches of n_features-dimensional
vectors.
        compiling (boolean): whether the model complied or not.
        opt (str, keras.optimizers): optimizer instance.
        loss (str, keras.losses): objective function.
        metrics (str, keras.metrics): List of metrics to be evaluated by the model during training an
d testing.
        drop_rate (float): dropout rate.

    Returns:
        model (keras.Model): a keras model.
    """

    model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(n_features,)),
        tf.keras.layers.Dense(n_units, activation='relu'),
        tf.keras.layers.Dropout(rate=drop_rate),
        tf.keras.layers.Dense(n_classes, activation='softmax')
    ])

    if compiling:

        if metrics is None or not isinstance(metrics, list):
            metrics = ['accuracy']

        model.compile(optimizer=opt, loss=loss, metrics=metrics)

    return model
```

In [0]:

```
#@title ## Create non-linear model

compiling = True #@param {type:"boolean"}

n_units = 5 #@param {type:"integer"}
opt = 'adam' #@param {type:"string"}
loss = 'categorical_crossentropy' #@param {type:"string"}
metrics = "accuracy" #@param ["accuracy"] {allow-input: true}
drop_rate = 0.2 #@param {type:"number"}

n_classes = y_train_c.shape[1] # 3, num_classes
n_features = x_train.shape[1] # 2, dimensions


nl_model_with_dropout = build_non_linear_with_dropout(n_classes, n_units, n_features, compiling, opt,
loss, metrics, drop_rate)
nl_model_with_dropout
```

Out[0]:

```
<tensorflow.python.keras.engine.sequential.Sequential at 0x7fa45163ffd0>
```

In [0]:
```
#@title ## Fit non-linear model with dropout.

r = nl_model_with_dropout.fit(x_train, y_train_c, validation_split=0.1, epochs=100, verbose=1)

history_dict = r.history
history_list = list(history_dict.keys())

print()
print('history_list', history_list)
```

```
Train on 904 samples, validate on 101 samples
Epoch 1/100
904/904 [==============================] - 0s 388us/sample - loss: 1.0544 - accuracy: 0.3639 - val_l
oss: 1.0434 - val_accuracy: 0.4158
Epoch 2/100
904/904 [==============================] - 0s 107us/sample - loss: 1.0366 - accuracy: 0.4004 - val_l
oss: 1.0260 - val_accuracy: 0.5050
Epoch 3/100
904/904 [==============================] - 0s 105us/sample - loss: 1.0250 - accuracy: 0.4856 - val_l
oss: 1.0095 - val_accuracy: 0.6238
Epoch 4/100
904/904 [==============================] - 0s 105us/sample - loss: 1.0135 - accuracy: 0.5199 - val_l
oss: 0.9937 - val_accuracy: 0.5941
Epoch 5/100
904/904 [==============================] - 0s 110us/sample - loss: 1.0002 - accuracy: 0.5155 - val_l
oss: 0.9781 - val_accuracy: 0.5941
Epoch 6/100
904/904 [==============================] - 0s 115us/sample - loss: 0.9814 - accuracy: 0.5221 - val_l
oss: 0.9631 - val_accuracy: 0.5743
Epoch 7/100
904/904 [==============================] - 0s 111us/sample - loss: 0.9748 - accuracy: 0.5299 - val_l
oss: 0.9484 - val_accuracy: 0.5644
Epoch 8/100
904/904 [==============================] - 0s 105us/sample - loss: 0.9572 - accuracy: 0.5188 - val_l
oss: 0.9356 - val_accuracy: 0.5545
Epoch 9/100
904/904 [==============================] - 0s 110us/sample - loss: 0.9533 - accuracy: 0.5044 - val_l
oss: 0.9216 - val_accuracy: 0.5644
Epoch 10/100
904/904 [==============================] - 0s 123us/sample - loss: 0.9417 - accuracy: 0.5133 - val_l
oss: 0.9082 - val_accuracy: 0.5545
Epoch 11/100
904/904 [==============================] - 0s 111us/sample - loss: 0.9240 - accuracy: 0.5398 - val_l
oss: 0.8960 - val_accuracy: 0.5545
Epoch 12/100
904/904 [==============================] - 0s 111us/sample - loss: 0.9174 - accuracy: 0.5243 - val_l
oss: 0.8843 - val_accuracy: 0.5050
Epoch 13/100
904/904 [==============================] - 0s 110us/sample - loss: 0.9043 - accuracy: 0.5542 - val_l
oss: 0.8736 - val_accuracy: 0.5248
Epoch 14/100
904/904 [==============================] - 0s 115us/sample - loss: 0.9005 - accuracy: 0.5542 - val_l
oss: 0.8632 - val_accuracy: 0.5347
```

```
Epoch 15/100
904/904 [==============================] - 0s 123us/sample - loss: 0.8944 - accuracy: 0.5254 - val_l
oss: 0.8533 - val_accuracy: 0.5347
Epoch 16/100
904/904 [==============================] - 0s 110us/sample - loss: 0.8728 - accuracy: 0.5619 - val_l
oss: 0.8446 - val_accuracy: 0.5446
Epoch 17/100
904/904 [==============================] - 0s 116us/sample - loss: 0.8864 - accuracy: 0.5299 - val_l
oss: 0.8362 - val_accuracy: 0.5545
Epoch 18/100
904/904 [==============================] - 0s 114us/sample - loss: 0.8753 - accuracy: 0.5299 - val_l
oss: 0.8281 - val_accuracy: 0.5644
Epoch 19/100
904/904 [==============================] - 0s 113us/sample - loss: 0.8708 - accuracy: 0.5409 - val_l
oss: 0.8219 - val_accuracy: 0.5743
Epoch 20/100
904/904 [==============================] - 0s 97us/sample - loss: 0.8780 - accuracy: 0.5321 - val_lo
ss: 0.8157 - val_accuracy: 0.5248
Epoch 21/100
904/904 [==============================] - 0s 105us/sample - loss: 0.8488 - accuracy: 0.5498 - val_l
oss: 0.8102 - val_accuracy: 0.5842
Epoch 22/100
904/904 [==============================] - 0s 101us/sample - loss: 0.8569 - accuracy: 0.5498 - val_l
oss: 0.8041 - val_accuracy: 0.5941
Epoch 23/100
904/904 [==============================] - 0s 98us/sample - loss: 0.8629 - accuracy: 0.5354 - val_lo
ss: 0.7998 - val_accuracy: 0.5941
Epoch 24/100
904/904 [==============================] - 0s 98us/sample - loss: 0.8633 - accuracy: 0.5376 - val_lo
ss: 0.7938 - val_accuracy: 0.5941
Epoch 25/100
904/904 [==============================] - 0s 98us/sample - loss: 0.8431 - accuracy: 0.5476 - val_lo
ss: 0.7886 - val_accuracy: 0.6040
Epoch 26/100
904/904 [==============================] - 0s 104us/sample - loss: 0.8546 - accuracy: 0.5387 - val_l
oss: 0.7830 - val_accuracy: 0.6238
Epoch 27/100
904/904 [==============================] - 0s 125us/sample - loss: 0.8314 - accuracy: 0.5564 - val_l
oss: 0.7768 - val_accuracy: 0.6139
Epoch 28/100
904/904 [==============================] - 0s 98us/sample - loss: 0.8433 - accuracy: 0.5420 - val_lo
ss: 0.7723 - val_accuracy: 0.6139
Epoch 29/100
```

```
904/904 [==============================] - 0s 99us/sample - loss: 0.8357 - accuracy: 0.5409 - val_lo
ss: 0.7689 - val_accuracy: 0.6337
Epoch 30/100
904/904 [==============================] - 0s 114us/sample - loss: 0.8287 - accuracy: 0.5509 - val_l
oss: 0.7647 - val_accuracy: 0.6238
Epoch 31/100
904/904 [==============================] - 0s 101us/sample - loss: 0.8367 - accuracy: 0.5487 - val_l
oss: 0.7605 - val_accuracy: 0.6436
Epoch 32/100
904/904 [==============================] - 0s 106us/sample - loss: 0.8306 - accuracy: 0.5420 - val_l
oss: 0.7562 - val_accuracy: 0.6337
Epoch 33/100
904/904 [==============================] - 0s 106us/sample - loss: 0.8327 - accuracy: 0.5409 - val_l
oss: 0.7523 - val_accuracy: 0.6337
Epoch 34/100
904/904 [==============================] - 0s 110us/sample - loss: 0.8265 - accuracy: 0.5509 - val_l
oss: 0.7493 - val_accuracy: 0.6337
Epoch 35/100
904/904 [==============================] - 0s 101us/sample - loss: 0.8137 - accuracy: 0.5686 - val_l
oss: 0.7446 - val_accuracy: 0.6436
Epoch 36/100
904/904 [==============================] - 0s 98us/sample - loss: 0.8086 - accuracy: 0.5631 - val_lo
ss: 0.7405 - val_accuracy: 0.6436
Epoch 37/100
904/904 [==============================] - 0s 101us/sample - loss: 0.8112 - accuracy: 0.5686 - val_l
oss: 0.7350 - val_accuracy: 0.6337
Epoch 38/100
904/904 [==============================] - 0s 103us/sample - loss: 0.8004 - accuracy: 0.5796 - val_l
oss: 0.7306 - val_accuracy: 0.6436
Epoch 39/100
904/904 [==============================] - 0s 105us/sample - loss: 0.8003 - accuracy: 0.5785 - val_l
oss: 0.7282 - val_accuracy: 0.6337
Epoch 40/100
904/904 [==============================] - 0s 113us/sample - loss: 0.8067 - accuracy: 0.5686 - val_l
oss: 0.7262 - val_accuracy: 0.6535
Epoch 41/100
904/904 [==============================] - 0s 98us/sample - loss: 0.7986 - accuracy: 0.5642 - val_lo
ss: 0.7237 - val_accuracy: 0.6733
Epoch 42/100
904/904 [==============================] - 0s 94us/sample - loss: 0.8007 - accuracy: 0.5708 - val_lo
ss: 0.7208 - val_accuracy: 0.6634
Epoch 43/100
904/904 [==============================] - 0s 96us/sample - loss: 0.8005 - accuracy: 0.5631 - val_lo
```

```
ss: 0.7171 - val_accuracy: 0.6634
Epoch 44/100
904/904 [==============================] - 0s 96us/sample - loss: 0.7974 - accuracy: 0.5487 - val_lo
ss: 0.7147 - val_accuracy: 0.6634
Epoch 45/100
904/904 [==============================] - 0s 112us/sample - loss: 0.7790 - accuracy: 0.5564 - val_l
oss: 0.7122 - val_accuracy: 0.6634
Epoch 46/100
904/904 [==============================] - 0s 105us/sample - loss: 0.7903 - accuracy: 0.5608 - val_l
oss: 0.7089 - val_accuracy: 0.6535
Epoch 47/100
904/904 [==============================] - 0s 99us/sample - loss: 0.7884 - accuracy: 0.5741 - val_lo
ss: 0.7075 - val_accuracy: 0.6436
Epoch 48/100
904/904 [==============================] - 0s 100us/sample - loss: 0.7860 - accuracy: 0.5542 - val_l
oss: 0.7051 - val_accuracy: 0.6535
Epoch 49/100
904/904 [==============================] - 0s 98us/sample - loss: 0.7820 - accuracy: 0.5564 - val_lo
ss: 0.7016 - val_accuracy: 0.6535
Epoch 50/100
904/904 [==============================] - 0s 99us/sample - loss: 0.7874 - accuracy: 0.5420 - val_lo
ss: 0.6991 - val_accuracy: 0.6535
Epoch 51/100
904/904 [==============================] - 0s 110us/sample - loss: 0.7833 - accuracy: 0.5608 - val_l
oss: 0.6963 - val_accuracy: 0.6535
Epoch 52/100
904/904 [==============================] - 0s 106us/sample - loss: 0.7828 - accuracy: 0.5553 - val_l
oss: 0.6939 - val_accuracy: 0.6634
Epoch 53/100
904/904 [==============================] - 0s 105us/sample - loss: 0.7799 - accuracy: 0.5642 - val_l
oss: 0.6924 - val_accuracy: 0.6634
Epoch 54/100
904/904 [==============================] - 0s 105us/sample - loss: 0.7832 - accuracy: 0.5299 - val_l
oss: 0.6904 - val_accuracy: 0.6634
Epoch 55/100
904/904 [==============================] - 0s 105us/sample - loss: 0.7645 - accuracy: 0.5365 - val_l
oss: 0.6893 - val_accuracy: 0.6634
Epoch 56/100
904/904 [==============================] - 0s 101us/sample - loss: 0.7704 - accuracy: 0.5708 - val_l
oss: 0.6873 - val_accuracy: 0.6634
Epoch 57/100
904/904 [==============================] - 0s 100us/sample - loss: 0.7720 - accuracy: 0.5520 - val_l
oss: 0.6848 - val_accuracy: 0.6535
```

```
Epoch 58/100
904/904 [==============================] - 0s 99us/sample - loss: 0.7653 - accuracy: 0.5409 - val_lo
ss: 0.6828 - val_accuracy: 0.6634
Epoch 59/100
904/904 [==============================] - 0s 96us/sample - loss: 0.7673 - accuracy: 0.5420 - val_lo
ss: 0.6803 - val_accuracy: 0.6733
Epoch 60/100
904/904 [==============================] - 0s 97us/sample - loss: 0.7609 - accuracy: 0.5376 - val_lo
ss: 0.6799 - val_accuracy: 0.6733
Epoch 61/100
904/904 [==============================] - 0s 98us/sample - loss: 0.7618 - accuracy: 0.5476 - val_lo
ss: 0.6780 - val_accuracy: 0.6733
Epoch 62/100
904/904 [==============================] - 0s 109us/sample - loss: 0.7676 - accuracy: 0.5487 - val_l
oss: 0.6765 - val_accuracy: 0.6535
Epoch 63/100
904/904 [==============================] - 0s 103us/sample - loss: 0.7668 - accuracy: 0.5254 - val_l
oss: 0.6747 - val_accuracy: 0.6733
Epoch 64/100
904/904 [==============================] - 0s 112us/sample - loss: 0.7554 - accuracy: 0.5487 - val_l
oss: 0.6726 - val_accuracy: 0.6535
Epoch 65/100
904/904 [==============================] - 0s 112us/sample - loss: 0.7539 - accuracy: 0.5863 - val_l
oss: 0.6723 - val_accuracy: 0.6931
Epoch 66/100
904/904 [==============================] - 0s 102us/sample - loss: 0.7428 - accuracy: 0.5553 - val_l
oss: 0.6689 - val_accuracy: 0.6535
Epoch 67/100
904/904 [==============================] - 0s 97us/sample - loss: 0.7568 - accuracy: 0.5719 - val_lo
ss: 0.6675 - val_accuracy: 0.6931
Epoch 68/100
904/904 [==============================] - 0s 107us/sample - loss: 0.7288 - accuracy: 0.6018 - val_l
oss: 0.6654 - val_accuracy: 0.6931
Epoch 69/100
904/904 [==============================] - 0s 100us/sample - loss: 0.7577 - accuracy: 0.5996 - val_l
oss: 0.6648 - val_accuracy: 0.7129
Epoch 70/100
904/904 [==============================] - 0s 104us/sample - loss: 0.7539 - accuracy: 0.5929 - val_l
oss: 0.6632 - val_accuracy: 0.7129
Epoch 71/100
904/904 [==============================] - 0s 104us/sample - loss: 0.7584 - accuracy: 0.6206 - val_l
oss: 0.6635 - val_accuracy: 0.7228
Epoch 72/100
```

```
904/904 [==============================] - 0s 115us/sample - loss: 0.7522 - accuracy: 0.6261 - val_l
oss: 0.6622 - val_accuracy: 0.7129
Epoch 73/100
904/904 [==============================] - 0s 97us/sample - loss: 0.7433 - accuracy: 0.6195 - val_lo
ss: 0.6603 - val_accuracy: 0.6733
Epoch 74/100
904/904 [==============================] - 0s 108us/sample - loss: 0.7595 - accuracy: 0.5885 - val_l
oss: 0.6605 - val_accuracy: 0.7030
Epoch 75/100
904/904 [==============================] - 0s 100us/sample - loss: 0.7390 - accuracy: 0.6051 - val_l
oss: 0.6588 - val_accuracy: 0.6337
Epoch 76/100
904/904 [==============================] - 0s 107us/sample - loss: 0.7676 - accuracy: 0.5675 - val_l
oss: 0.6609 - val_accuracy: 0.6832
Epoch 77/100
904/904 [==============================] - 0s 104us/sample - loss: 0.7346 - accuracy: 0.6106 - val_l
oss: 0.6587 - val_accuracy: 0.6436
Epoch 78/100
904/904 [==============================] - 0s 97us/sample - loss: 0.7360 - accuracy: 0.5852 - val_lo
ss: 0.6563 - val_accuracy: 0.6436
Epoch 79/100
904/904 [==============================] - 0s 96us/sample - loss: 0.7517 - accuracy: 0.5653 - val_lo
ss: 0.6560 - val_accuracy: 0.6634
Epoch 80/100
904/904 [==============================] - 0s 97us/sample - loss: 0.7457 - accuracy: 0.5752 - val_lo
ss: 0.6557 - val_accuracy: 0.6634
Epoch 81/100
904/904 [==============================] - 0s 98us/sample - loss: 0.7272 - accuracy: 0.5896 - val_lo
ss: 0.6544 - val_accuracy: 0.6634
Epoch 82/100
904/904 [==============================] - 0s 98us/sample - loss: 0.7406 - accuracy: 0.6029 - val_lo
ss: 0.6527 - val_accuracy: 0.6337
Epoch 83/100
904/904 [==============================] - 0s 116us/sample - loss: 0.7445 - accuracy: 0.5896 - val_l
oss: 0.6534 - val_accuracy: 0.6238
Epoch 84/100
904/904 [==============================] - 0s 99us/sample - loss: 0.7326 - accuracy: 0.5951 - val_lo
ss: 0.6528 - val_accuracy: 0.5941
Epoch 85/100
904/904 [==============================] - 0s 94us/sample - loss: 0.7355 - accuracy: 0.5885 - val_lo
ss: 0.6515 - val_accuracy: 0.6040
Epoch 86/100
904/904 [==============================] - 0s 97us/sample - loss: 0.7357 - accuracy: 0.5841 - val_lo
```

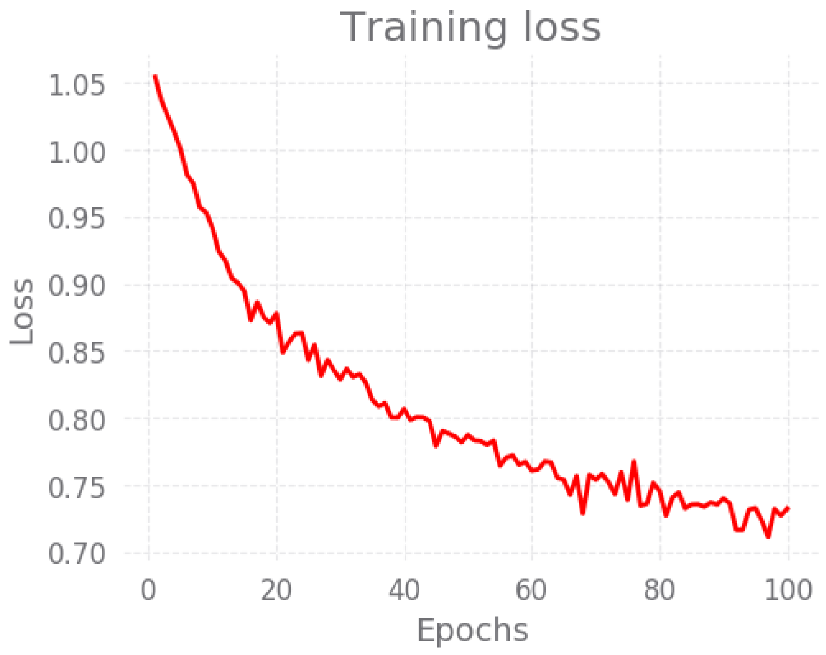
```
ss: 0.6515 - val_accuracy: 0.6139
Epoch 87/100
904/904 [==============================] - 0s 99us/sample - loss: 0.7338 - accuracy: 0.5885 - val_lo
ss: 0.6520 - val_accuracy: 0.5941
Epoch 88/100
904/904 [==============================] - 0s 96us/sample - loss: 0.7370 - accuracy: 0.5951 - val_lo
ss: 0.6499 - val_accuracy: 0.5941
Epoch 89/100
904/904 [==============================] - 0s 98us/sample - loss: 0.7354 - accuracy: 0.6018 - val_lo
ss: 0.6493 - val_accuracy: 0.5842
Epoch 90/100
904/904 [==============================] - 0s 102us/sample - loss: 0.7401 - accuracy: 0.5752 - val_l
oss: 0.6482 - val_accuracy: 0.6040
Epoch 91/100
904/904 [==============================] - 0s 104us/sample - loss: 0.7362 - accuracy: 0.5841 - val_l
oss: 0.6482 - val_accuracy: 0.5743
Epoch 92/100
904/904 [==============================] - 0s 100us/sample - loss: 0.7165 - accuracy: 0.5785 - val_l
oss: 0.6464 - val_accuracy: 0.6040
Epoch 93/100
904/904 [==============================] - 0s 96us/sample - loss: 0.7164 - accuracy: 0.6095 - val_lo
ss: 0.6439 - val_accuracy: 0.5941
Epoch 94/100
904/904 [==============================] - 0s 113us/sample - loss: 0.7317 - accuracy: 0.5785 - val_l
oss: 0.6446 - val_accuracy: 0.6139
Epoch 95/100
904/904 [==============================] - 0s 104us/sample - loss: 0.7328 - accuracy: 0.5918 - val_l
oss: 0.6446 - val_accuracy: 0.6139
Epoch 96/100
904/904 [==============================] - 0s 106us/sample - loss: 0.7235 - accuracy: 0.5808 - val_l
oss: 0.6429 - val_accuracy: 0.6337
Epoch 97/100
904/904 [==============================] - 0s 105us/sample - loss: 0.7112 - accuracy: 0.6162 - val_l
oss: 0.6411 - val_accuracy: 0.6139
Epoch 98/100
904/904 [==============================] - 0s 108us/sample - loss: 0.7322 - accuracy: 0.5653 - val_l
oss: 0.6396 - val_accuracy: 0.6337
Epoch 99/100
904/904 [==============================] - 0s 105us/sample - loss: 0.7272 - accuracy: 0.5808 - val_l
oss: 0.6393 - val_accuracy: 0.6535
Epoch 100/100
904/904 [==============================] - 0s 102us/sample - loss: 0.7324 - accuracy: 0.5863 - val_l
oss: 0.6414 - val_accuracy: 0.6337
```

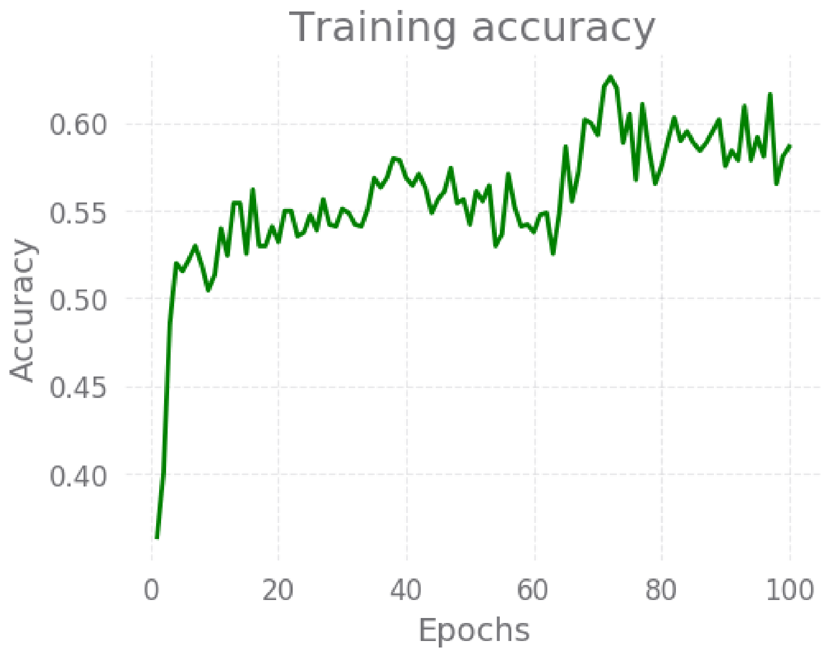
```
history_list ['loss', 'accuracy', 'val_loss', 'val_accuracy']
```

In [0]:
```
#@title ## Evaluating

evaluation = nl_model_with_dropout.evaluate(x_test, y_test_c, verbose=0)

print('Loss: %.3f\t Accuracy: %.2f%%' % (evaluation[0], evaluation[1] * 100))
```

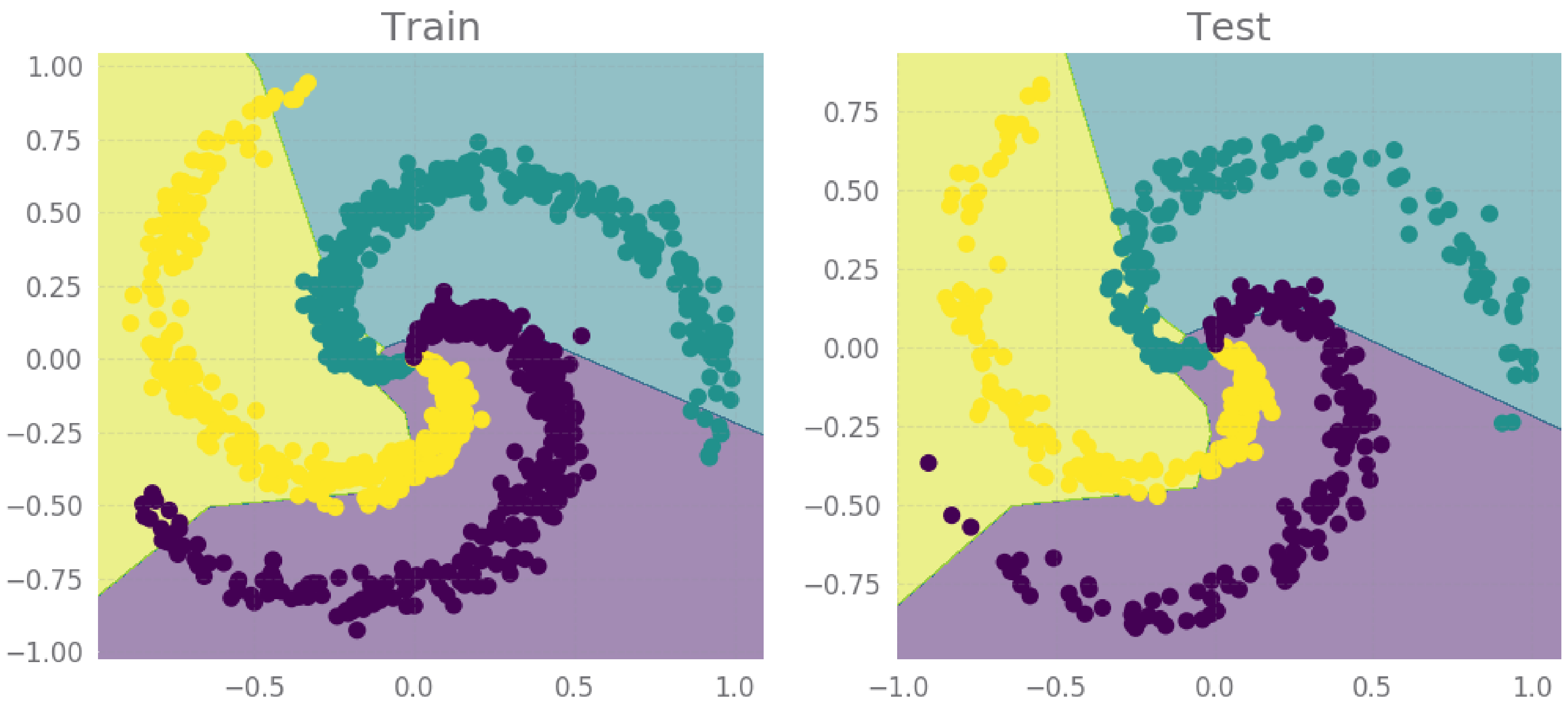
```
Loss: 0.656      Accuracy: 68.08%
```

In [0]:
```
#@title ## Plotting

xfig = 12.0 #@param {type:"number"}
yfig = 4.0 #@param {type:"number"}

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(xfig, yfig))


loss = history_dict[history_list[history_list.index('loss')]]
acc = history_dict[history_list[history_list.index('accuracy')]]

epochs = range(1, len(loss) + 1)


ax1.plot(epochs, loss, 'r', label='Training loss')
ax1.set_title('Training loss')
ax1.set_xlabel('Epochs')
ax1.set_ylabel('Loss')

ax2.plot(epochs, acc, 'g', label='Training accuracy')
ax2.set_title('Training accuracy')
ax2.set_xlabel('Epochs')
ax2.set_ylabel('Accuracy')

plt.show()
```


Training loss
Loss
Epochs
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
20
40
60
80
100

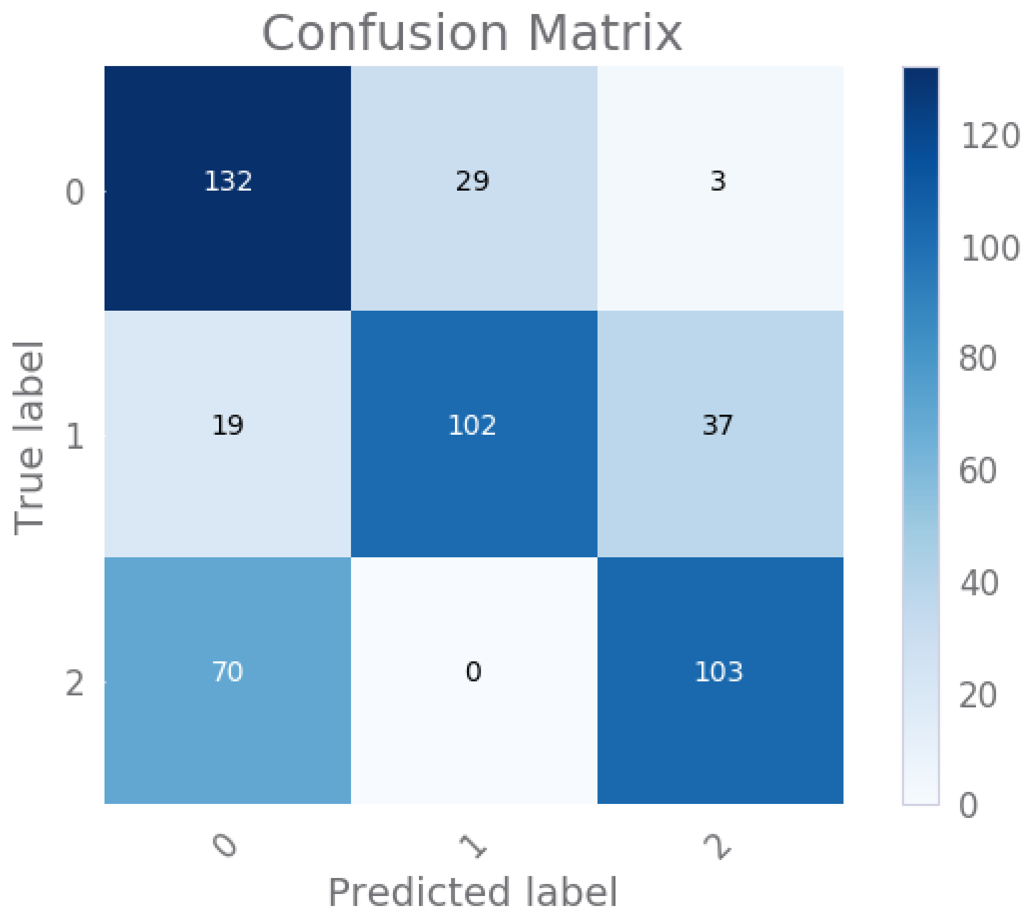

Training accuracy
Accuracy
Epochs
0.60
0.55
0.50
0.45
0.40
0
20
40
60
80
100

In [0]:
```
#@title ## Visualize the decision boundary

xfig = 12.0 #@param {type:"number"}
yfig = 5.0 #@param {type:"number"}

plt.figure(figsize=(xfig, yfig))
plt.subplot(1, 2, 1)
plt.title('Train')
plot_mc_decision_boundary(model=nl_model_with_dropout, x=x_train, y=y_train)
plt.subplot(1, 2, 2)
plt.title('Test')
plot_mc_decision_boundary(model=nl_model_with_dropout, x=x_test, y=y_test)
plt.show()
```



In [0]:
```
#@title ## Confusion matrix

pred_test = nl_model_with_dropout.predict(x_test)
pred_test = np.argmax(pred_test, axis=1)

cm = confusion_matrix(y_test, pred_test)
plot_confusion_matrix(cm=cm, classes=[0, 1, 2])

print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

           0       0.60      0.80      0.69       164
           1       0.78      0.65      0.71       158
           2       0.72      0.60      0.65       173

    accuracy                           0.68       495
   macro avg       0.70      0.68      0.68       495
weighted avg       0.70      0.68      0.68       495
```



Visualization

```
In [0]: #@title ## Build non-linear model [MLP Model](https://en.wikipedia.org/wiki/Multilayer_perceptron)

def build_non_linear_v2(n_classes, n_units, n_features, drop_rate=0.5):
    """ Build a linear model [MLP Model].

    Args:
        n_classes (int): dimensionality of the output space.
        n_units (int): dimensionality of the hidden layer.
        n_features (int): indicates that the expected input will be batches of n_features-dimensional
vectors.
        drop_rate (float): dropout rate.

    Returns:
        model (keras.Model): a keras model.
    """

    model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(n_features,)),
        tf.keras.layers.Dense(n_units, activation='relu'),
        tf.keras.layers.Dropout(rate=drop_rate),
        tf.keras.layers.Dense(n_units, activation='relu'),
        tf.keras.layers.Dropout(rate=drop_rate),
        tf.keras.layers.Dense(n_units, activation='relu'),
        tf.keras.layers.Dropout(rate=drop_rate),
        tf.keras.layers.Dense(n_classes, activation='softmax')
    ])

    return model
```

In [0]:
```
#@title ## Create non-linear model

compiling = True #@param {type:"boolean"}

n_units = 20 #@param {type:"integer"}
opt = 'adam' #@param {type:"string"}
loss = 'categorical_crossentropy' #@param {type:"string"}
metrics = "accuracy" #@param ["accuracy"] {allow-input: true}
drop_rate = 0.2 #@param {type:"number"}

n_classes = y_train_c.shape[1] # 3, num_classes
n_features = x_train.shape[1] # 2, dimensions


model = build_non_linear_v2(
    n_classes,
    n_units,
    n_features,
    drop_rate)

model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.Adam(lr=0.002),
    metrics=['accuracy'])
model
```

Out[0]:
```
<tensorflow.python.keras.engine.sequential.Sequential at 0x7fa44f99ca58>
```

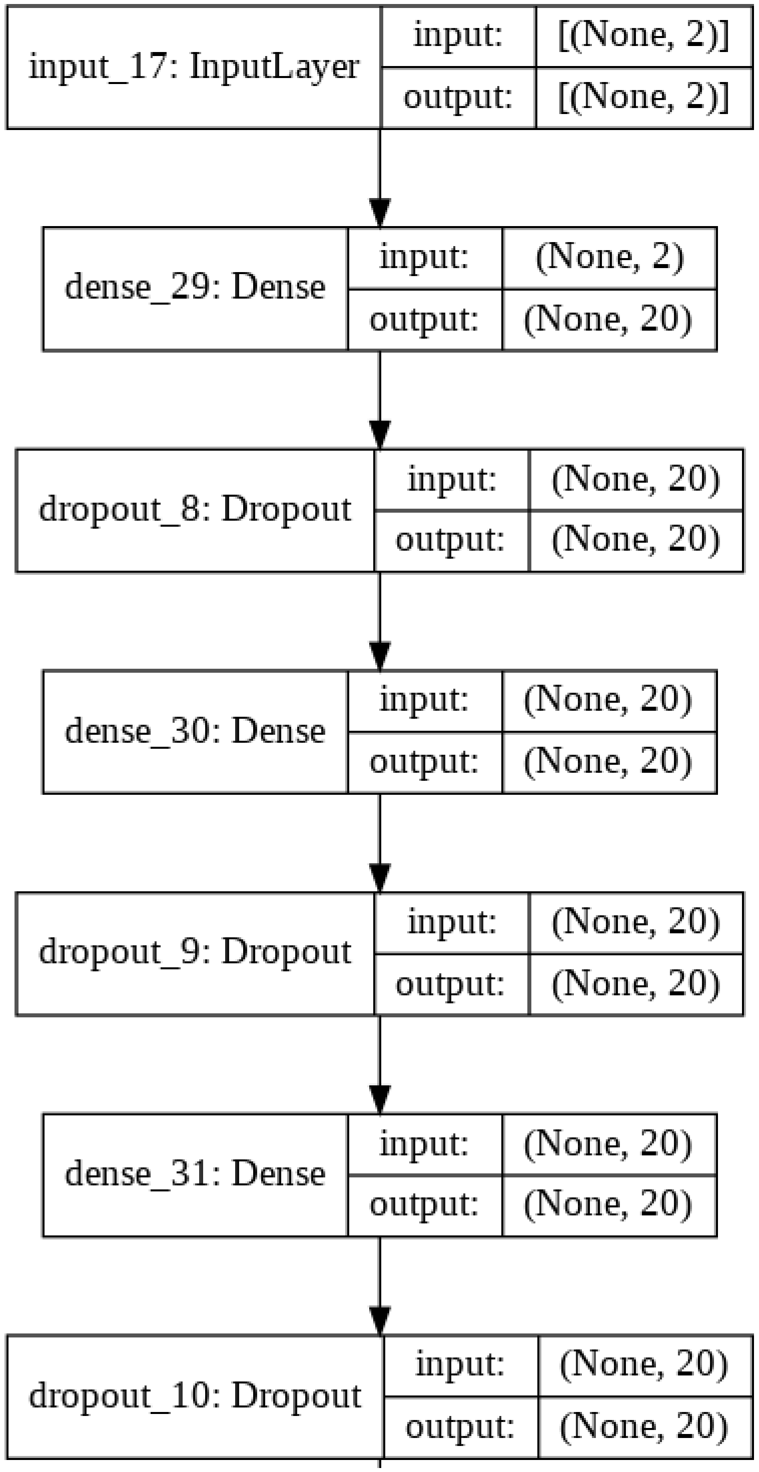
In [0]:
```
#@title # Summary of the model

model.summary()
```

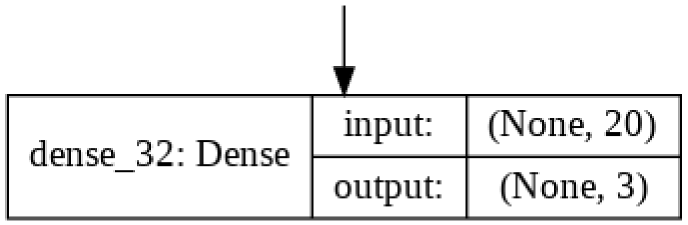
```
Model: "sequential_16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_29 (Dense)             (None, 20)                60        
_________________________________________________________________
dropout_8 (Dropout)          (None, 20)                0         
_________________________________________________________________
dense_30 (Dense)             (None, 20)                420       
_________________________________________________________________
dropout_9 (Dropout)          (None, 20)                0         
_________________________________________________________________
dense_31 (Dense)             (None, 20)                420       
_________________________________________________________________
dropout_10 (Dropout)         (None, 20)                0         
_________________________________________________________________
dense_32 (Dense)             (None, 3)                 63        
=================================================================
Total params: 963
Trainable params: 963
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

```
#@title ## Model Arch
from IPython.display import Image


outputs_dir = os.path.join(root_dir, 'outputs')
os.makedirs(outputs_dir, exist_ok=True)

to_file = os.path.join(outputs_dir, 'model-arch.png')
tf.keras.utils.plot_model(
    model,
    to_file=to_file,
    show_layer_names=True,
    show_shapes=True)


Image(filename=to_file)
```

Out[0]:


input_17: InputLayer
input: [(None, 2)]
output: [(None, 2)]
dense_29: Dense
input: (None, 2)
output: (None, 20)
dropout_8: Dropout
input: (None, 20)
output: (None, 20)
dense_30: Dense
input: (None, 20)
output: (None, 20)
dropout_9: Dropout
input: (None, 20)
output: (None, 20)
dense_31: Dense
input: (None, 20)
output: (None, 20)
dropout_10: Dropout
input: (None, 20)
output: (None, 20)



In [0]:

```
#@title ## Tensorboard callbacks

logdir = "logs/scalars/" + datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
tensorboard_callback
```

Out[0]:

```
<tensorflow.python.keras.callbacks.TensorBoard at 0x7fa44f698f60>
```

In [0]:
```
#@title ## Fit non-linear model.

r = model.fit(
    x_train, y_train_c,
    validation_split=0.1,
    batch_size=32,
    epochs=100,
    verbose=1,
    callbacks=[tensorboard_callback])

history_dict = r.history
history_list = list(history_dict.keys())

print()
print('history_list', history_list)
```

```
Train on 904 samples, validate on 101 samples
Epoch 1/100

W0719 09:07:26.078772 140347580262272 callbacks.py:241] Method (on_train_batch_end) is slow compared
to the batch update (0.243796). Check your callbacks.

 32/904 [>.............................] - ETA: 10s - loss: 1.0964 - accuracy: 0.2188

W0719 09:07:26.109556 140347580262272 callbacks.py:241] Method (on_train_batch_end) is slow compared
to the batch update (0.129082). Check your callbacks.
```

```
904/904 [==============================] - 1s 914us/sample - loss: 1.0289 - accuracy: 0.3761 - val_l
oss: 0.9290 - val_accuracy: 0.5248
Epoch 2/100
904/904 [==============================] - 0s 179us/sample - loss: 0.9213 - accuracy: 0.5564 - val_l
oss: 0.8288 - val_accuracy: 0.5545
Epoch 3/100
904/904 [==============================] - 0s 172us/sample - loss: 0.8349 - accuracy: 0.6073 - val_l
oss: 0.7403 - val_accuracy: 0.6733
Epoch 4/100
904/904 [==============================] - 0s 163us/sample - loss: 0.7531 - accuracy: 0.6350 - val_l
oss: 0.6508 - val_accuracy: 0.7426
Epoch 5/100
904/904 [==============================] - 0s 163us/sample - loss: 0.7247 - accuracy: 0.6493 - val_l
oss: 0.6049 - val_accuracy: 0.7426
Epoch 6/100
904/904 [==============================] - 0s 181us/sample - loss: 0.6500 - accuracy: 0.7058 - val_l
oss: 0.5566 - val_accuracy: 0.7723
Epoch 7/100
904/904 [==============================] - 0s 167us/sample - loss: 0.6268 - accuracy: 0.7179 - val_l
oss: 0.5039 - val_accuracy: 0.7822
Epoch 8/100
904/904 [==============================] - 0s 160us/sample - loss: 0.5893 - accuracy: 0.7367 - val_l
oss: 0.4634 - val_accuracy: 0.7624
Epoch 9/100
904/904 [==============================] - 0s 195us/sample - loss: 0.5508 - accuracy: 0.7478 - val_l
oss: 0.4224 - val_accuracy: 0.8020
Epoch 10/100
904/904 [==============================] - 0s 158us/sample - loss: 0.5260 - accuracy: 0.7832 - val_l
oss: 0.3914 - val_accuracy: 0.8218
Epoch 11/100
904/904 [==============================] - 0s 157us/sample - loss: 0.4735 - accuracy: 0.7954 - val_l
oss: 0.3413 - val_accuracy: 0.9010
Epoch 12/100
904/904 [==============================] - 0s 160us/sample - loss: 0.4781 - accuracy: 0.7876 - val_l
oss: 0.3240 - val_accuracy: 0.9208
Epoch 13/100
904/904 [==============================] - 0s 157us/sample - loss: 0.4175 - accuracy: 0.8341 - val_l
oss: 0.2874 - val_accuracy: 0.9505
Epoch 14/100
904/904 [==============================] - 0s 162us/sample - loss: 0.4294 - accuracy: 0.8341 - val_l
oss: 0.2740 - val_accuracy: 0.9505
Epoch 15/100
904/904 [==============================] - 0s 183us/sample - loss: 0.3896 - accuracy: 0.8429 - val_l
```

```
oss: 0.2604 - val_accuracy: 0.9406
Epoch 16/100
904/904 [==============================] - 0s 162us/sample - loss: 0.3974 - accuracy: 0.8462 - val_l
oss: 0.2428 - val_accuracy: 0.9505
Epoch 17/100
904/904 [==============================] - 0s 163us/sample - loss: 0.3730 - accuracy: 0.8750 - val_l
oss: 0.2273 - val_accuracy: 0.9505
Epoch 18/100
904/904 [==============================] - 0s 176us/sample - loss: 0.3548 - accuracy: 0.8606 - val_l
oss: 0.2099 - val_accuracy: 0.9604
Epoch 19/100
904/904 [==============================] - 0s 166us/sample - loss: 0.3521 - accuracy: 0.8662 - val_l
oss: 0.1919 - val_accuracy: 0.9604
Epoch 20/100
904/904 [==============================] - 0s 167us/sample - loss: 0.3375 - accuracy: 0.8794 - val_l
oss: 0.1896 - val_accuracy: 0.9604
Epoch 21/100
904/904 [==============================] - 0s 163us/sample - loss: 0.3343 - accuracy: 0.8761 - val_l
oss: 0.1913 - val_accuracy: 0.9505
Epoch 22/100
904/904 [==============================] - 0s 185us/sample - loss: 0.3309 - accuracy: 0.8916 - val_l
oss: 0.1857 - val_accuracy: 0.9505
Epoch 23/100
904/904 [==============================] - 0s 161us/sample - loss: 0.3060 - accuracy: 0.8827 - val_l
oss: 0.1653 - val_accuracy: 0.9604
Epoch 24/100
904/904 [==============================] - 0s 170us/sample - loss: 0.2689 - accuracy: 0.8938 - val_l
oss: 0.1477 - val_accuracy: 0.9604
Epoch 25/100
904/904 [==============================] - 0s 168us/sample - loss: 0.3107 - accuracy: 0.8861 - val_l
oss: 0.1362 - val_accuracy: 0.9604
Epoch 26/100
904/904 [==============================] - 0s 160us/sample - loss: 0.2773 - accuracy: 0.9082 - val_l
oss: 0.1375 - val_accuracy: 0.9604
Epoch 27/100
904/904 [==============================] - 0s 166us/sample - loss: 0.2990 - accuracy: 0.8916 - val_l
oss: 0.1512 - val_accuracy: 0.9703
Epoch 28/100
904/904 [==============================] - 0s 163us/sample - loss: 0.2445 - accuracy: 0.9148 - val_l
oss: 0.1358 - val_accuracy: 0.9703
Epoch 29/100
904/904 [==============================] - 0s 172us/sample - loss: 0.2422 - accuracy: 0.9270 - val_l
oss: 0.1221 - val_accuracy: 0.9604
```

```
Epoch 30/100
904/904 [==============================] - 0s 161us/sample - loss: 0.2113 - accuracy: 0.9259 - val_l
oss: 0.1171 - val_accuracy: 0.9604
Epoch 31/100
904/904 [==============================] - 0s 168us/sample - loss: 0.2760 - accuracy: 0.8960 - val_l
oss: 0.1232 - val_accuracy: 0.9802
Epoch 32/100
904/904 [==============================] - 0s 169us/sample - loss: 0.2488 - accuracy: 0.9226 - val_l
oss: 0.1101 - val_accuracy: 0.9703
Epoch 33/100
904/904 [==============================] - 0s 160us/sample - loss: 0.2629 - accuracy: 0.9004 - val_l
oss: 0.1192 - val_accuracy: 0.9604
Epoch 34/100
904/904 [==============================] - 0s 170us/sample - loss: 0.2454 - accuracy: 0.9148 - val_l
oss: 0.1109 - val_accuracy: 0.9703
Epoch 35/100
904/904 [==============================] - 0s 180us/sample - loss: 0.2430 - accuracy: 0.9215 - val_l
oss: 0.1125 - val_accuracy: 0.9802
Epoch 36/100
904/904 [==============================] - 0s 161us/sample - loss: 0.2294 - accuracy: 0.9303 - val_l
oss: 0.1052 - val_accuracy: 0.9703
Epoch 37/100
904/904 [==============================] - 0s 163us/sample - loss: 0.2306 - accuracy: 0.9192 - val_l
oss: 0.1105 - val_accuracy: 0.9802
Epoch 38/100
904/904 [==============================] - 0s 161us/sample - loss: 0.1911 - accuracy: 0.9414 - val_l
oss: 0.0979 - val_accuracy: 0.9802
Epoch 39/100
904/904 [==============================] - 0s 158us/sample - loss: 0.2416 - accuracy: 0.9137 - val_l
oss: 0.0987 - val_accuracy: 0.9802
Epoch 40/100
904/904 [==============================] - 0s 162us/sample - loss: 0.2055 - accuracy: 0.9314 - val_l
oss: 0.1030 - val_accuracy: 0.9604
Epoch 41/100
904/904 [==============================] - 0s 158us/sample - loss: 0.1936 - accuracy: 0.9392 - val_l
oss: 0.0922 - val_accuracy: 0.9802
Epoch 42/100
904/904 [==============================] - 0s 178us/sample - loss: 0.1934 - accuracy: 0.9281 - val_l
oss: 0.0927 - val_accuracy: 0.9802
Epoch 43/100
904/904 [==============================] - 0s 164us/sample - loss: 0.2037 - accuracy: 0.9281 - val_l
oss: 0.0966 - val_accuracy: 0.9703
Epoch 44/100
```

```
904/904 [==============================] - 0s 165us/sample - loss: 0.1841 - accuracy: 0.9381 - val_l
oss: 0.0905 - val_accuracy: 0.9703
Epoch 45/100
904/904 [==============================] - 0s 163us/sample - loss: 0.1744 - accuracy: 0.9414 - val_l
oss: 0.0825 - val_accuracy: 0.9802
Epoch 46/100
904/904 [==============================] - 0s 162us/sample - loss: 0.1606 - accuracy: 0.9546 - val_l
oss: 0.0912 - val_accuracy: 0.9703
Epoch 47/100
904/904 [==============================] - 0s 172us/sample - loss: 0.1656 - accuracy: 0.9381 - val_l
oss: 0.0795 - val_accuracy: 0.9703
Epoch 48/100
904/904 [==============================] - 0s 164us/sample - loss: 0.1847 - accuracy: 0.9403 - val_l
oss: 0.0821 - val_accuracy: 0.9703
Epoch 49/100
904/904 [==============================] - 0s 178us/sample - loss: 0.2082 - accuracy: 0.9192 - val_l
oss: 0.0849 - val_accuracy: 0.9604
Epoch 50/100
904/904 [==============================] - 0s 172us/sample - loss: 0.1856 - accuracy: 0.9314 - val_l
oss: 0.0813 - val_accuracy: 0.9604
Epoch 51/100
904/904 [==============================] - 0s 167us/sample - loss: 0.1798 - accuracy: 0.9469 - val_l
oss: 0.0808 - val_accuracy: 0.9802
Epoch 52/100
904/904 [==============================] - 0s 166us/sample - loss: 0.1985 - accuracy: 0.9325 - val_l
oss: 0.0866 - val_accuracy: 0.9703
Epoch 53/100
904/904 [==============================] - 0s 167us/sample - loss: 0.1982 - accuracy: 0.9281 - val_l
oss: 0.0775 - val_accuracy: 0.9802
Epoch 54/100
904/904 [==============================] - 0s 161us/sample - loss: 0.1767 - accuracy: 0.9347 - val_l
oss: 0.0890 - val_accuracy: 0.9703
Epoch 55/100
904/904 [==============================] - 0s 174us/sample - loss: 0.1794 - accuracy: 0.9381 - val_l
oss: 0.0869 - val_accuracy: 0.9703
Epoch 56/100
904/904 [==============================] - 0s 169us/sample - loss: 0.1898 - accuracy: 0.9358 - val_l
oss: 0.0888 - val_accuracy: 0.9703
Epoch 57/100
904/904 [==============================] - 0s 166us/sample - loss: 0.1704 - accuracy: 0.9458 - val_l
oss: 0.0750 - val_accuracy: 0.9703
Epoch 58/100
904/904 [==============================] - 0s 162us/sample - loss: 0.1308 - accuracy: 0.9558 - val_l
```

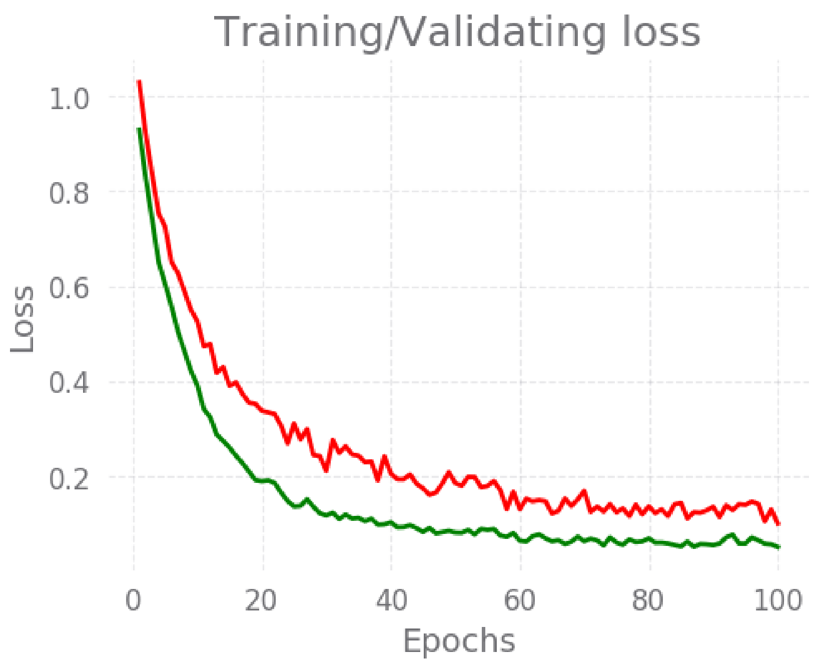
```
oss: 0.0721 - val_accuracy: 0.9703
Epoch 59/100
904/904 [==============================] - 0s 156us/sample - loss: 0.1674 - accuracy: 0.9469 - val_l
oss: 0.0805 - val_accuracy: 0.9703
Epoch 60/100
904/904 [==============================] - 0s 157us/sample - loss: 0.1305 - accuracy: 0.9668 - val_l
oss: 0.0639 - val_accuracy: 0.9802
Epoch 61/100
904/904 [==============================] - 0s 165us/sample - loss: 0.1524 - accuracy: 0.9480 - val_l
oss: 0.0623 - val_accuracy: 0.9802
Epoch 62/100
904/904 [==============================] - 0s 177us/sample - loss: 0.1471 - accuracy: 0.9546 - val_l
oss: 0.0736 - val_accuracy: 0.9604
Epoch 63/100
904/904 [==============================] - 0s 164us/sample - loss: 0.1495 - accuracy: 0.9513 - val_l
oss: 0.0775 - val_accuracy: 0.9703
Epoch 64/100
904/904 [==============================] - 0s 166us/sample - loss: 0.1471 - accuracy: 0.9513 - val_l
oss: 0.0692 - val_accuracy: 0.9703
Epoch 65/100
904/904 [==============================] - 0s 170us/sample - loss: 0.1215 - accuracy: 0.9657 - val_l
oss: 0.0627 - val_accuracy: 0.9703
Epoch 66/100
904/904 [==============================] - 0s 160us/sample - loss: 0.1271 - accuracy: 0.9624 - val_l
oss: 0.0651 - val_accuracy: 0.9703
Epoch 67/100
904/904 [==============================] - 0s 162us/sample - loss: 0.1530 - accuracy: 0.9491 - val_l
oss: 0.0573 - val_accuracy: 0.9802
Epoch 68/100
904/904 [==============================] - 0s 152us/sample - loss: 0.1369 - accuracy: 0.9546 - val_l
oss: 0.0618 - val_accuracy: 0.9802
Epoch 69/100
904/904 [==============================] - 0s 163us/sample - loss: 0.1509 - accuracy: 0.9546 - val_l
oss: 0.0729 - val_accuracy: 0.9703
Epoch 70/100
904/904 [==============================] - 0s 157us/sample - loss: 0.1691 - accuracy: 0.9469 - val_l
oss: 0.0630 - val_accuracy: 0.9802
Epoch 71/100
904/904 [==============================] - 0s 154us/sample - loss: 0.1247 - accuracy: 0.9602 - val_l
oss: 0.0685 - val_accuracy: 0.9604
Epoch 72/100
904/904 [==============================] - 0s 156us/sample - loss: 0.1360 - accuracy: 0.9591 - val_l
oss: 0.0647 - val_accuracy: 0.9703
```

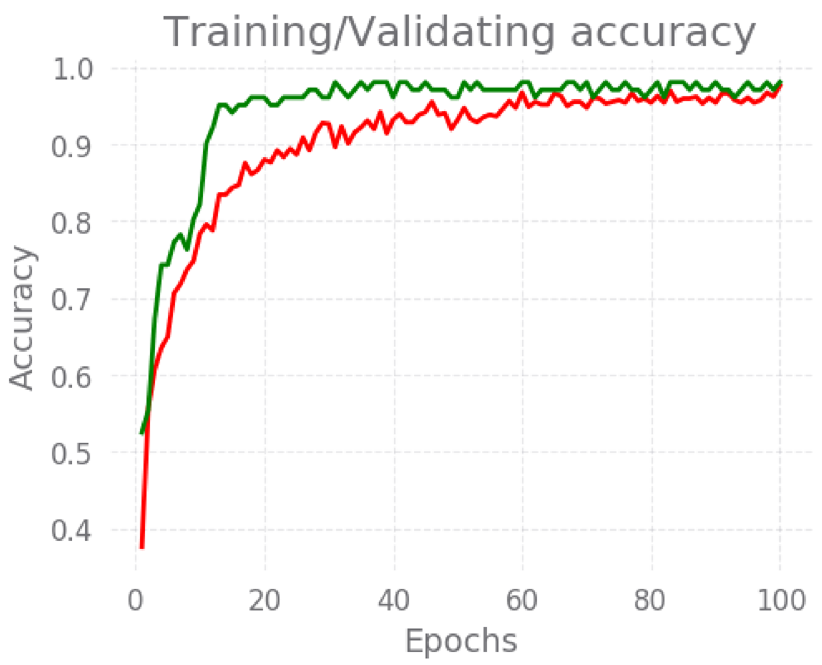
```
Epoch 73/100
904/904 [==============================] - 0s 181us/sample - loss: 0.1253 - accuracy: 0.9524 - val_l
oss: 0.0543 - val_accuracy: 0.9802
Epoch 74/100
904/904 [==============================] - 0s 166us/sample - loss: 0.1411 - accuracy: 0.9546 - val_l
oss: 0.0713 - val_accuracy: 0.9703
Epoch 75/100
904/904 [==============================] - 0s 163us/sample - loss: 0.1237 - accuracy: 0.9569 - val_l
oss: 0.0602 - val_accuracy: 0.9703
Epoch 76/100
904/904 [==============================] - 0s 171us/sample - loss: 0.1322 - accuracy: 0.9535 - val_l
oss: 0.0557 - val_accuracy: 0.9802
Epoch 77/100
904/904 [==============================] - 0s 170us/sample - loss: 0.1160 - accuracy: 0.9657 - val_l
oss: 0.0669 - val_accuracy: 0.9703
Epoch 78/100
904/904 [==============================] - 0s 159us/sample - loss: 0.1399 - accuracy: 0.9558 - val_l
oss: 0.0611 - val_accuracy: 0.9703
Epoch 79/100
904/904 [==============================] - 0s 156us/sample - loss: 0.1198 - accuracy: 0.9591 - val_l
oss: 0.0626 - val_accuracy: 0.9604
Epoch 80/100
904/904 [==============================] - 0s 153us/sample - loss: 0.1366 - accuracy: 0.9535 - val_l
oss: 0.0692 - val_accuracy: 0.9703
Epoch 81/100
904/904 [==============================] - 0s 162us/sample - loss: 0.1224 - accuracy: 0.9624 - val_l
oss: 0.0600 - val_accuracy: 0.9802
Epoch 82/100
904/904 [==============================] - 0s 151us/sample - loss: 0.1301 - accuracy: 0.9535 - val_l
oss: 0.0602 - val_accuracy: 0.9604
Epoch 83/100
904/904 [==============================] - 0s 169us/sample - loss: 0.1161 - accuracy: 0.9690 - val_l
oss: 0.0584 - val_accuracy: 0.9802
Epoch 84/100
904/904 [==============================] - 0s 153us/sample - loss: 0.1406 - accuracy: 0.9546 - val_l
oss: 0.0552 - val_accuracy: 0.9802
Epoch 85/100
904/904 [==============================] - 0s 157us/sample - loss: 0.1436 - accuracy: 0.9591 - val_l
oss: 0.0520 - val_accuracy: 0.9802
Epoch 86/100
904/904 [==============================] - 0s 159us/sample - loss: 0.1112 - accuracy: 0.9591 - val_l
oss: 0.0624 - val_accuracy: 0.9703
Epoch 87/100
```

```
904/904 [==============================] - 0s 163us/sample - loss: 0.1243 - accuracy: 0.9613 - val_l
oss: 0.0512 - val_accuracy: 0.9802
Epoch 88/100
904/904 [==============================] - 0s 158us/sample - loss: 0.1232 - accuracy: 0.9524 - val_l
oss: 0.0573 - val_accuracy: 0.9703
Epoch 89/100
904/904 [==============================] - 0s 168us/sample - loss: 0.1280 - accuracy: 0.9602 - val_l
oss: 0.0566 - val_accuracy: 0.9703
Epoch 90/100
904/904 [==============================] - 0s 163us/sample - loss: 0.1356 - accuracy: 0.9535 - val_l
oss: 0.0549 - val_accuracy: 0.9802
Epoch 91/100
904/904 [==============================] - 0s 159us/sample - loss: 0.1142 - accuracy: 0.9657 - val_l
oss: 0.0583 - val_accuracy: 0.9703
Epoch 92/100
904/904 [==============================] - 0s 156us/sample - loss: 0.1386 - accuracy: 0.9657 - val_l
oss: 0.0710 - val_accuracy: 0.9703
Epoch 93/100
904/904 [==============================] - 0s 164us/sample - loss: 0.1286 - accuracy: 0.9569 - val_l
oss: 0.0768 - val_accuracy: 0.9604
Epoch 94/100
904/904 [==============================] - 0s 154us/sample - loss: 0.1409 - accuracy: 0.9535 - val_l
oss: 0.0584 - val_accuracy: 0.9703
Epoch 95/100
904/904 [==============================] - 0s 157us/sample - loss: 0.1395 - accuracy: 0.9602 - val_l
oss: 0.0583 - val_accuracy: 0.9802
Epoch 96/100
904/904 [==============================] - 0s 172us/sample - loss: 0.1467 - accuracy: 0.9535 - val_l
oss: 0.0711 - val_accuracy: 0.9703
Epoch 97/100
904/904 [==============================] - 0s 180us/sample - loss: 0.1415 - accuracy: 0.9569 - val_l
oss: 0.0649 - val_accuracy: 0.9703
Epoch 98/100
904/904 [==============================] - 0s 160us/sample - loss: 0.1049 - accuracy: 0.9668 - val_l
oss: 0.0580 - val_accuracy: 0.9802
Epoch 99/100
904/904 [==============================] - 0s 170us/sample - loss: 0.1299 - accuracy: 0.9613 - val_l
oss: 0.0566 - val_accuracy: 0.9703
Epoch 100/100
904/904 [==============================] - 0s 162us/sample - loss: 0.1006 - accuracy: 0.9757 - val_l
oss: 0.0510 - val_accuracy: 0.9802

history_list ['loss', 'accuracy', 'val_loss', 'val_accuracy']
```

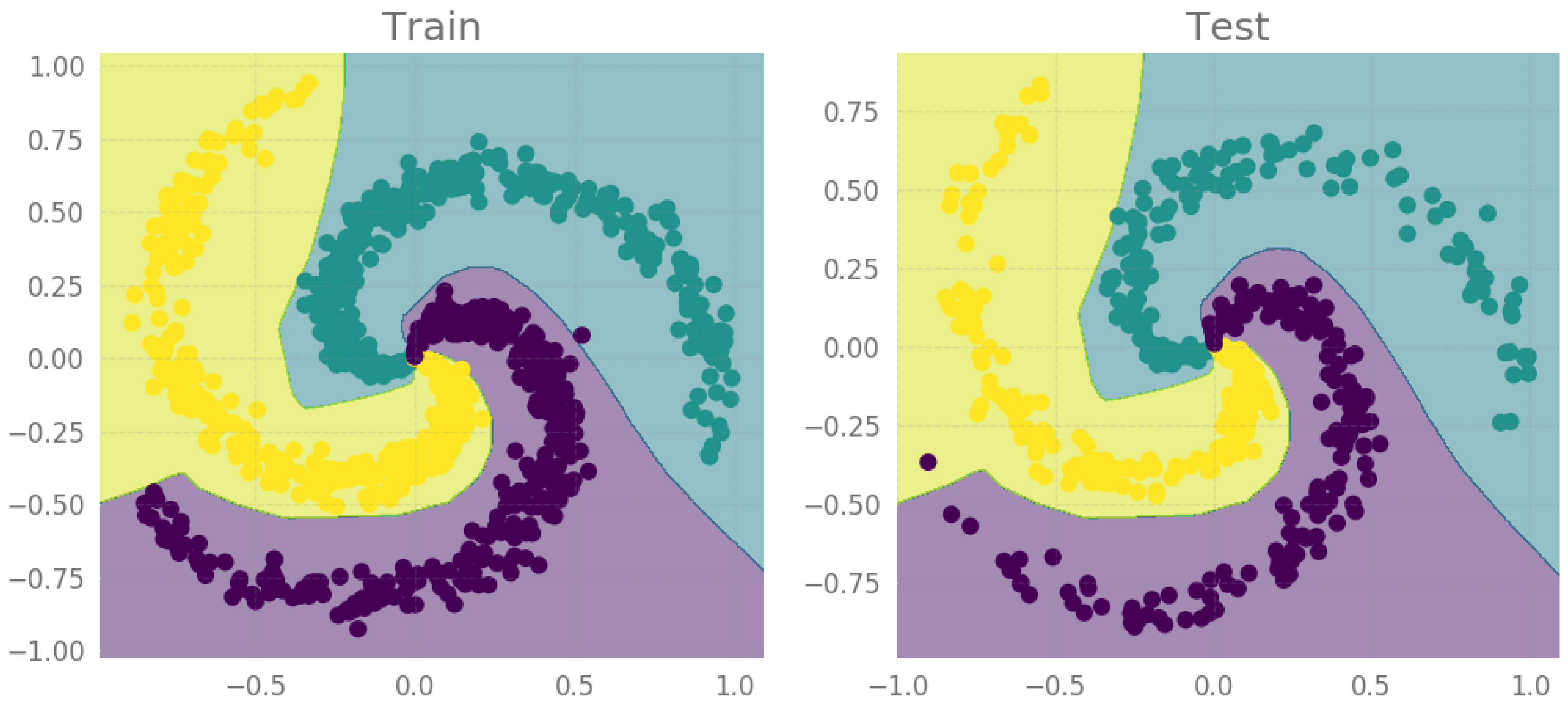
In [0]:

```
#@title ## Plotting

xfig = 12.0 #@param {type:"number"}
yfig = 4.0 #@param {type:"number"}

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(xfig, yfig))


loss = history_dict[history_list[history_list.index('loss')]]
val_loss = history_dict[history_list[history_list.index('val_loss')]]

acc = history_dict[history_list[history_list.index('accuracy')]]
val_acc = history_dict[history_list[history_list.index('val_accuracy')]]

epochs = range(1, len(loss) + 1)


ax1.plot(epochs, loss, 'r', label='Training loss')
ax1.plot(epochs, val_loss, 'g', label='Validating loss')
ax1.set_title('Training/Validating loss')
ax1.set_xlabel('Epochs')
ax1.set_ylabel('Loss')

ax2.plot(epochs, acc, 'r', label='Training accuracy')
ax2.plot(epochs, val_acc, 'g', label='Validating accuracy')
ax2.set_title('Training/Validating accuracy')
ax2.set_xlabel('Epochs')
ax2.set_ylabel('Accuracy')

plt.show()
```


Training/Validating loss
Loss
Epochs
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100

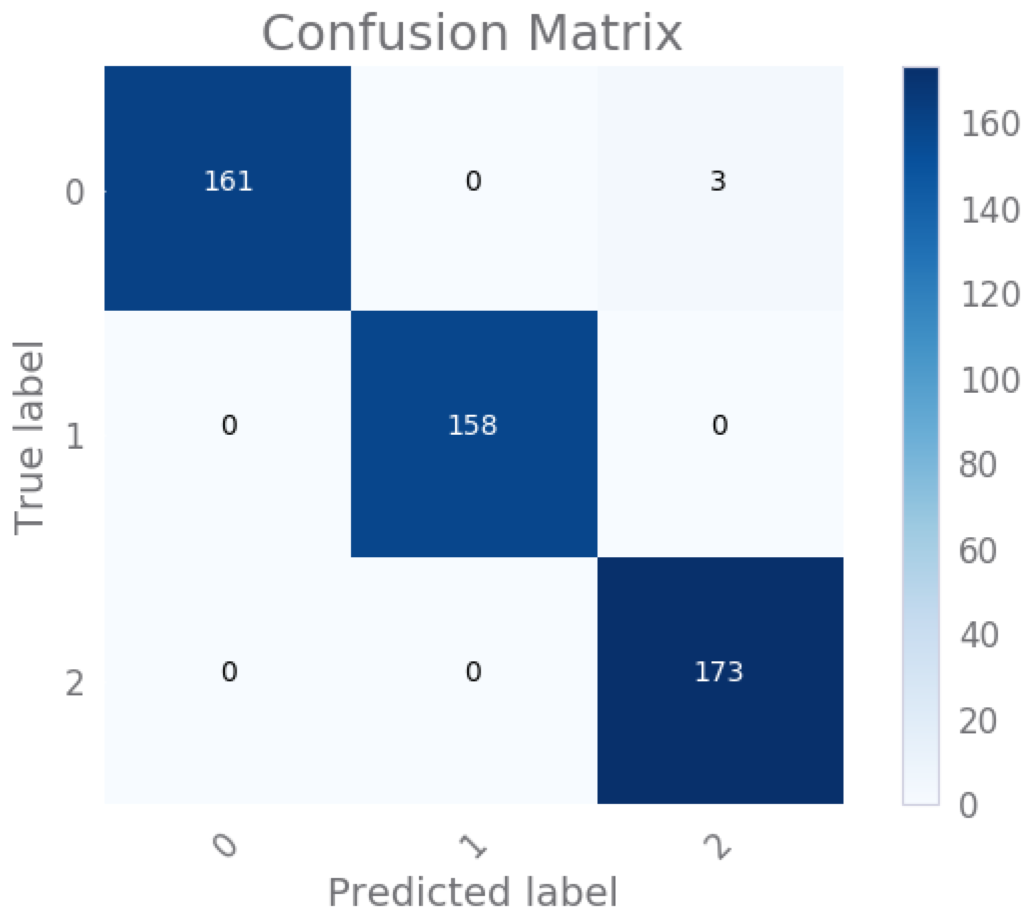

Training/Validating accuracy
Accuracy
Epochs
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
20
40
60
80
100

```
In [0]: #@title ## Visualize the decision boundary

        xfig = 12.0 #@param {type:"number"}
        yfig = 5.0 #@param {type:"number"}

        plt.figure(figsize=(xfig, yfig))
        plt.subplot(1, 2, 1)
        plt.title('Train')
        plot_mc_decision_boundary(model=model, x=x_train, y=y_train)
        plt.subplot(1, 2, 2)
        plt.title('Test')
        plot_mc_decision_boundary(model=model, x=x_test, y=y_test)
        plt.show()
```



In [0]:
```
#@title ## Confusion matrix

pred_test = model.predict(x_test)
pred_test = np.argmax(pred_test, axis=1)

cm = confusion_matrix(y_test, pred_test)
plot_confusion_matrix(cm=cm, classes=[0, 1, 2])

print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       164
           1       1.00      1.00      1.00       158
           2       0.98      1.00      0.99       173

    accuracy                           0.99       495
   macro avg       0.99      0.99      0.99       495
weighted avg       0.99      0.99      0.99       495
```



Tensorboard

In [0]:
```
%load_ext tensorboard
```

In [0]:
```
%tensorboard --logdir logs/scalars
```

In [0]:
```
```