# final-project

August 7, 2020

## 1 Data Mining Final Project: Brain Tumor Classification


#### 1.0.1 Sanjay Jars

#### 1.0.2 DSC 550

#### 1.0.3 Bellevue University


With every year, the number of patients with a brain tumor is increasing. There are two classes of brain tumors, primary and secondary tumors. Primary tumors have several types; one of the frequently found is a meningioma type. This type of tumors found near the top and outer curve of the brain. Meningioma is slowly growing noncancerous tumors that cause seizures and visual problems. Glioma is an abnormal growth in glial cells present around the neurons in the brain. Pituitary tumors grow in pituitary glands that affect body functions. It is a very difficult to locate, detect, and select the infected tumor portion in the brain from the MRI (Magnetic resonance images). This tedious and time-consuming task is performed by radiologists and medical field experts. The accuracy of this task is mainly dependent on the experience and expertise of the person performing this task. So, if we use a machine learning model to perform this task, it will help to overcome the shortcomings with the person involved in performing this task. So, I think if we can automate this process of classifying the tumors by using machine learning algorithms, it will improve the accuracy of the results and cost due to the expertise required. If we correctly classify the tumors, the specific treatment to that type of tumor can be applied. The accurate information about type and location helps in planning the surgical process for its removal. I will be following an approach involving image preprocessing like noise removal, cropping image, extraction, and classification. The segmentation will be used to segment and divide the tumor region. The extraction will be used to extract information from the segmented image. Classification will help to compare the extracted information with information from the available dataset. Image pre-processing will be performed to reduce the noise level in the image. For segmentation, I am thinking about using K-means unsupervised algorithm. Then we can use a Support vector machine that can be used to classify the segmented image. Once the model is trained, we can give new images and the model will predict the class for the new image. I time permits I would also like to create a CNN model for prediction. I will be using the brain tumor images dataset from https://figshare.com/articles/brain_tumor_dataset/1512427. This dataset contains 3064 T1-weighted contrast-enhanced images from 233 patients with three kinds of brain tumors. The data is organized in the MATLAB data format (.mat file). Each file stores a struct containing the following fields for an image: cjdata.label: 1 for meningioma, 2 for glioma, 3 for pituitary tumor cjdata.PID: patient ID cjdata.image: image data cjdata.tumorBorder: a vector storing the coordinates of discrete points on tumor border. cjdata.tumorMask: a binary image with 1s indicating tumor region. I am planning to measure the accuracy of classification by testing the model with

the test data set. The accuracy of the model can be majored by using the confusion matrix.

https://pdfs.semanticscholar.org/b70e/4a5455a4531ca650272474d49d29e5e1da5c.pdf
https://figshare.com/articles/brain_tumor_dataset/1512427

[1]:
```
import concurrent.futures as cf
import os
import random
import time

import cv2
import  h5py1 as h5  # library to load HDF5 binary file format
import keras
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.ndimage
import seaborn as sns
import sklearn
import tensorflow
from keras.layers import (
    Activation,
    BatchNormalization,
    Conv2D,
    Dense,
    Flatten,
    Input,
    MaxPool2D,
    ZeroPadding2D,
)
from keras.models import Sequential
from keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
from matplotlib import __version__ as mpv
from sklearn.model_selection import train_test_split

print("Using version %s of cv2" % cv2.__version__)
print("Using version %s of pandas" % pd.__version__)
print("Using version %s of matplotlib" % mpv)
print("Using version %s of seaborn" % sns.__version__)
print("Using version %s of sklearn" % sklearn.__version__)
print("Using version %s of numpy" % np.__version__)
print("Using version %s of h5py" % h5.__version__)
print("Using version %s of keras" % keras.__version__)
print("Using version %s of tensorflow" % tensorflow.__version__)
print("Using version %s of scipy" % scipy.__version__)
```

```
Using version 4.2.0 of cv2
Using version 1.0.5 of pandas
```

```
Using version 3.2.2 of matplotlib
Using version 0.10.1 of seaborn
Using version 0.23.1 of sklearn
Using version 1.18.5 of numpy
Using version 2.10.0 of h5py
Using version 2.4.3 of keras
Using version 2.2.0 of tensorflow
Using version 1.4.1 of scipy
```

### 1.0.4 Configurations

[2]:
```
seed = 13
tensorflow.random.set_seed(seed)
np.random.seed(seed)
random.seed(seed)
folder = "original-mat-files"
target_folder = "converted-images/"  # plt.style.use("seaborn-darkgrid")
```

### 1.0.5 Scan Source .mat files

[3]:
```
onlyfiles = [
    os.path.join(folder, f)
    for f in os.listdir(folder)
    if os.path.isfile(os.path.join(folder, f)) and f.endswith(".mat")
]
os.makedirs(target_folder, exist_ok=True)
```

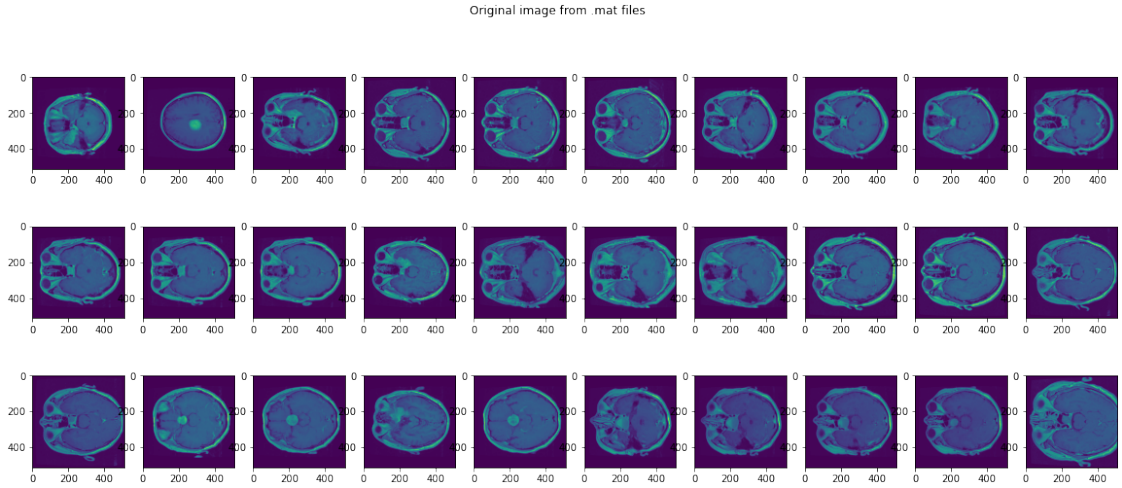
### 1.0.6 Utility class to parse the .mat file and read contents image and labels

[4]:
```
class MatImage(object):
    patient_id = ""
    image = ""
    label = ""
    tumor_border = ""
    tumor_mask = ""
    file_name = ""

    def __init__(self, file_name, file_path=None):
        path = file_name
        if file_path is not None:
            path = os.path.join(file_path, file_name)
        f = h5.File(path, "a")

        self.file_name = file_name
        self.image = np.mat(f["/cjdata/image"])
        self.patient_id = np.array(f["/cjdata/PID"])
        self.label = np.array(f["/cjdata/label"])
```

```
        self.tumor_border = np.mat(f["/cjdata/tumorBorder"])
        self.tumor_mask = np.mat(f["/cjdata/tumorMask"])

    # function to plot image with mask
    def draw_image_with_mask(self):
        ax = sns.heatmap(self.tumor_mask, alpha=0.2)
        ax = sns.heatmap(self.image)
        plt.show()

    # Function to apply medain filter on image for removing noise
    def get_median_filtered_image(self, filter_size=2):
        return scipy.ndimage.median_filter(self.image, filter_size)

    def tostring(self):
        result = "Patient Id:"
        for pid in self.patient_id:
            result += str(id) + ","
        result += " label"
        for label in self.label:
            result += str(label) + ","
        result += "File Name:" + self.file_name
        return result
```

### 1.0.7 Display Original images

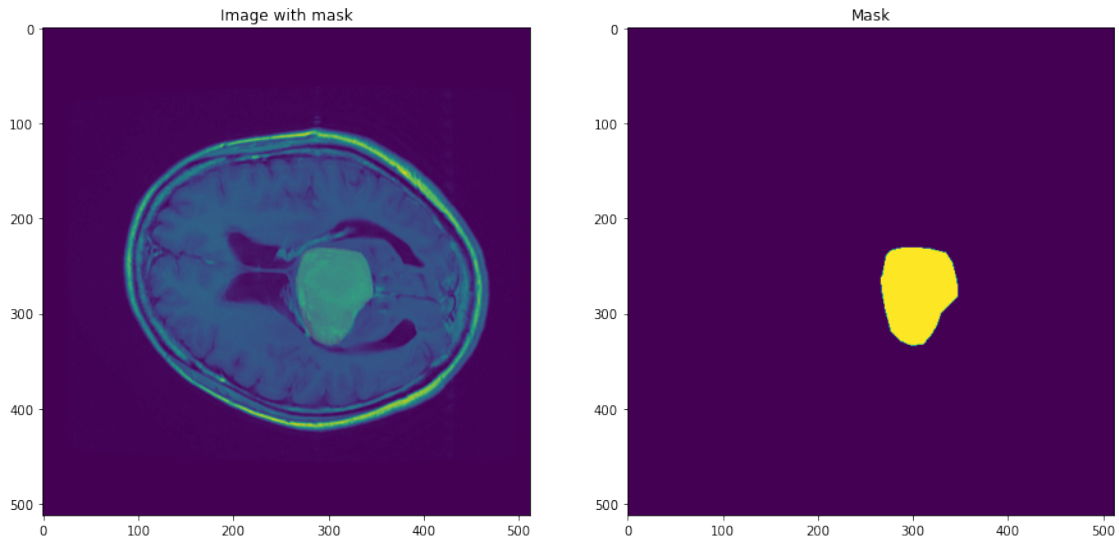
[5]:
```
plt.rcParams["figure.figsize"] = [20, 8]
fig, axs = plt.subplots(3, 10)
i = 0
for filepath in onlyfiles:
    p1 = MatImage(onlyfiles[i])
    axs[int(i / 10), i % 10].imshow(p1.image)
    i += 1
    if i >= 30:
        break
plt.suptitle("Original image from .mat files")
plt.show()
del p1
```




### 1.0.8 Image with mask plotted on top of image

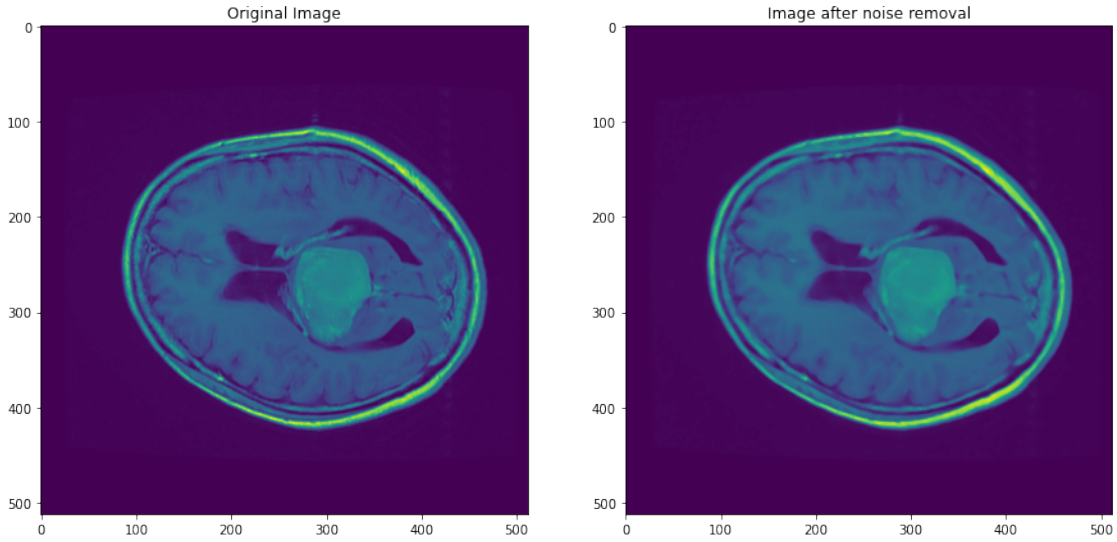
This just for information this is not used in this model

[6]:
```
p1 = MatImage(folder + "/22.mat")
plt.rcParams["figure.figsize"] = [15, 8]
fig, axs = plt.subplots(1, 2)
axs[0].imshow(p1.image)
axs[0].imshow(p1.tumor_mask, alpha=0.1)
axs[0].set(title="Image with mask")
axs[1].imshow(p1.tumor_mask)
axs[1].set(title="Mask")
plt.show()
```



### 1.0.9 Remove noise by applying median filter

[7]:
```
p1 = MatImage(folder + "/22.mat")
plt.rcParams["figure.figsize"] = [15, 8]
fig, axs = plt.subplots(1, 2)
axs[0].imshow(p1.image)
axs[0].set(title="Original Image")
axs[1].imshow(p1.get_median_filtered_image(5))
axs[1].set(title="Image after noise removal")
plt.show()
```




### 1.0.10 Read mat files

### 1.0.11 1. Use median filter to enhance images

### 1.0.12 2. Crop the images to only contain actual brain image and remove extra image areas

### 1.0.13 3. Resize all images to 256*256 dimension

[8]:
```
IMG_SIZE = (256, 256)


def crop_image(image_path, IMG_SIZE=(256, 256)):
    # read image
    img = cv2.imread(image_path)
    # resize image to smaller dimensions
    img = cv2.resize(img, dsize=IMG_SIZE, interpolation=cv2.INTER_CUBIC)
```

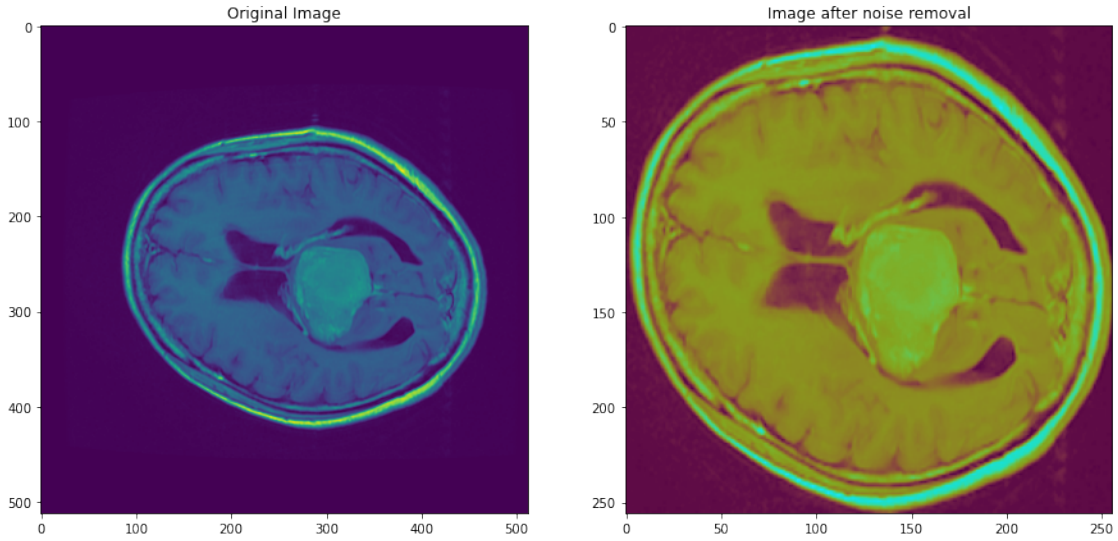
```
    # convert to gray
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    # add some blurring
    gray = cv2.GaussianBlur(gray, (5, 5), 0)

    # threshold the image, then perform a series of erosions +
    # dilations to remove any small regions of noise
    thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
    thresh = cv2.erode(thresh, None, iterations=2)
    thresh = cv2.dilate(thresh, None, iterations=2)

    # find contours in thresholded image, then grab the largest one
    cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.
↪CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    c = max(cnts, key=cv2.contourArea)

    # find the extreme points
    extLeft = tuple(c[c[:, :, 0].argmin()][0])
    extRight = tuple(c[c[:, :, 0].argmax()][0])
    extTop = tuple(c[c[:, :, 1].argmin()][0])
    extBot = tuple(c[c[:, :, 1].argmax()][0])

    # add contour on the image
    img_cnt = cv2.drawContours(img.copy(), [c], -1, (0, 255, 255), 4)

    # add extreme points
    img_pnt = cv2.circle(img_cnt.copy(), extLeft, 8, (0, 0, 255), -1)
    img_pnt = cv2.circle(img_pnt, extRight, 8, (0, 255, 0), -1)
    img_pnt = cv2.circle(img_pnt, extTop, 8, (255, 0, 0), -1)
    img_pnt = cv2.circle(img_pnt, extBot, 8, (255, 255, 0), -1)

    # crop
    ADD_PIXELS = 0
    new_img = img[
        extTop[1] - ADD_PIXELS : extBot[1] + ADD_PIXELS,
        extLeft[0] - ADD_PIXELS : extRight[0] + ADD_PIXELS,
    ].copy()
    # resize image to input dimensions after cropping
    new_img = cv2.resize(new_img, dsize=IMG_SIZE, interpolation=cv2.INTER_CUBIC)
    # return gray scale image
    gray = cv2.cvtColor(new_img, cv2.COLOR_RGB2GRAY)
    return gray
```

### 1.0.14 Function to save image as png

[9]:
```
def save_image(image_array, file_path, format="png"):
    fig, ax = plt.subplots()
    im = ax.imshow(image_array)
    plt.imsave(
        file_path, image_array, format=format,
    )
    plt.close(fig)
```

### 1.0.15 Crop Image and convert to grayscale

[10]:
```
p1 = MatImage(folder + "/22.mat")
plt.rcParams["figure.figsize"] = [15, 8]
fig, axs = plt.subplots(1, 2)
axs[0].imshow(p1.image)
axs[0].set(title="Original Image")
axs[1].imshow(cv2.imread(target_folder + "/22_1.png"))
axs[1].set(title="Image after noise removal")
plt.show()
```




### 1.0.16 Transform files as mentioned above and save them to folder

[ ]:
```
def save_filese_as_image(mat_file_path):
    # read input .mat file
    mat_image = MatImage(mat_file_path)
    filename = mat_image.file_name.split("/")[-1].split(".")[0]
    target_file_path = target_folder + filename + "_" + str(int(mat_image.
↪label[0][0])) + ".png"
```

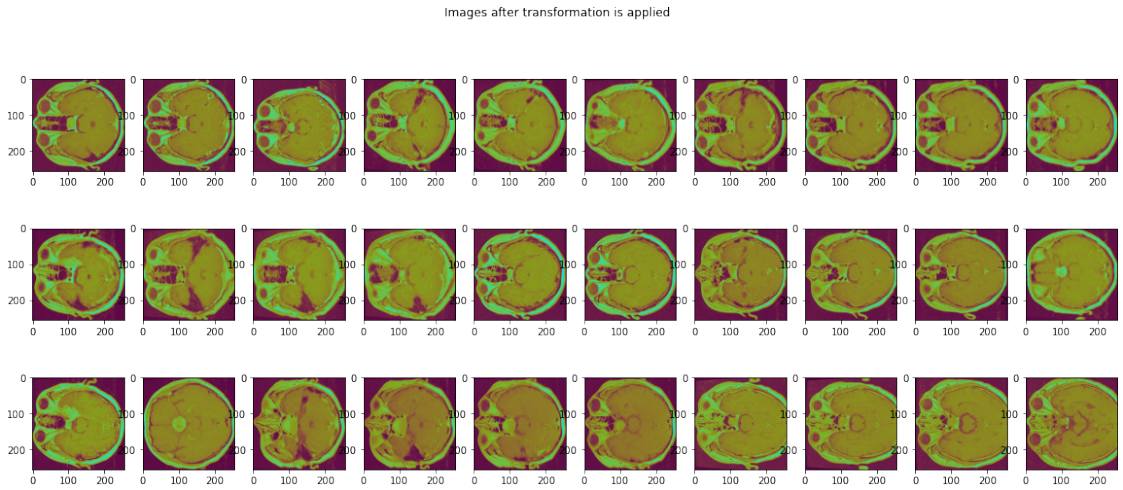
```
    # save image after applying median filter
    save_image(mat_image.get_median_filtered_image(), target_file_path)
    # apply cropping
    conv_image = crop_image(target_file_path)
    # save image again
    save_image(conv_image, target_file_path)

# use multiple cores for above operation
start_time = time.time()
with cf.ProcessPoolExecutor() as executor:
        results = executor.map(save_filese_as_image, onlyfiles)
print(f"--- {(time.time() - start_time)} seconds for transforming␣
 ↪{len(onlyfiles)} files---"  )
```

### 1.0.17 Scan all converyed image files

[11]:
```
onlyfiles = [
    os.path.join(target_folder, f)
    for f in os.listdir(target_folder)
    if os.path.isfile(os.path.join(target_folder, f)) and f.endswith(".png")
]
```

### 1.0.18 Images after above transformations

[12]:
```
plt.rcParams["figure.figsize"] = [20, 8]
fig, axs = plt.subplots(3, 10)
i = 0
for filepath in onlyfiles:
    img = cv2.imread(filepath)
    axs[int(i / 10), i % 10].imshow(img)
    i += 1
    if i >= 30:
        break
plt.suptitle("Images after transformation is applied")
plt.show()
del img
```




### 1.0.19 Function to load all images into numpy arrays with labels

[13]:
```
def load_image(image_path):
    img = cv2.imread(image_path)
    filename, label = get_filename_label(image_path)
    return img, label, filename


def get_filename_label(image_path):
    filename = image_path.split("/")[-1].split(".")[0]
    label = filename.split("_")[-1]
    return filename, int(label)


def get_one_hot_encoded_label(label):
    if label == 1:
        return [1, 0, 0]
    elif label == 2:
        return [0, 1, 0]
    else:
        return [0, 0, 1]


def get_data_set():
    images = []
    labels = []

    filenames = []
    with cf.ProcessPoolExecutor() as executor:
        results = executor.map(load_image, onlyfiles)
```

```
    for result in results:
        if result[0] is not None:
            if result[0].shape[0] != 256 or result[0].shape[1] != 256:
                print("Dropping image as dimensions are not correct", result[2])
            else:
                image = result[0]
                labels.append(get_one_hot_encoded_label(result[1]))
                images.append(cv2.cvtColor(image, cv2.COLOR_RGB2GRAY))
                filenames.append(result[2])
    output_image = np.empty((len(images), 256, 256))

    for i in range(len(images)):
        output_image[i] = images[i]

    return output_image, np.array(labels)
```

### 1.0.20 Load all images

[14]:
```
start_time = time.time()
dataset_xc, dataset_yc = get_data_set()
print("Shape of dataset-X", dataset_xc.shape)
print("Shape of dataset-Y", dataset_yc.shape)
print(
    f"--- {(time.time() - start_time)} seconds for loading {len(dataset_xc)}␣
 ↪files---"
)
```

```
Shape of dataset-X (3064, 256, 256)
Shape of dataset-Y (3064, 3)
--- 7.7368245124816895 seconds for loading 3064 files---
```

### 1.0.21 Add augmented images

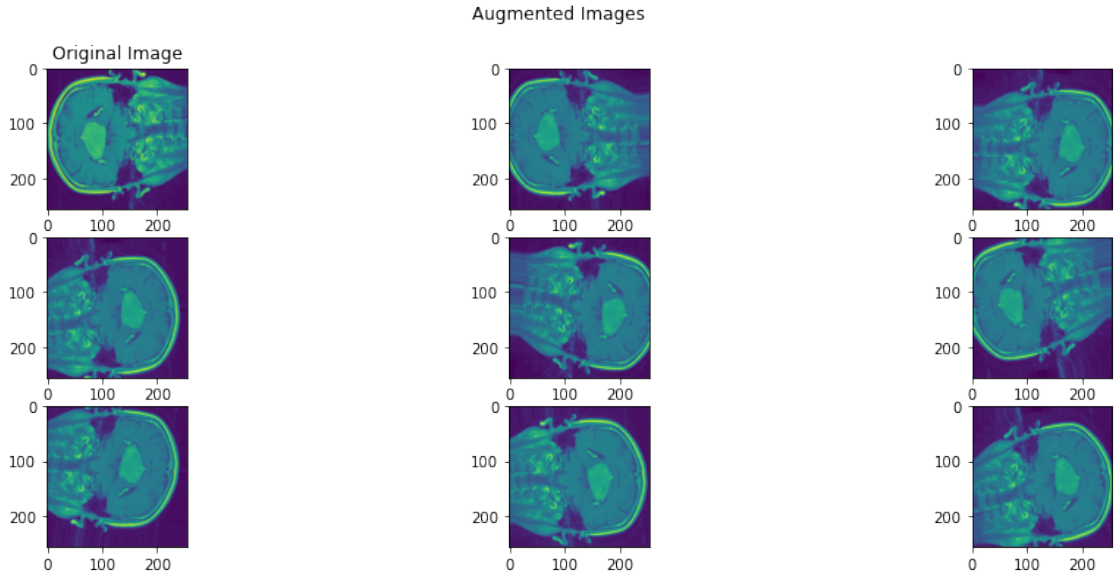
[15]:
```
def augment_data(file_path, no_of_samples=1):
    imgs = []
    labels = []
    img = load_img(file_path)
    # convert to numpy array
    data = img_to_array(img)
    # expand dimension to one sample
    samples = np.expand_dims(data, 0)
    data_gen = ImageDataGenerator(
        rotation_range=10,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.1,
```

```
        brightness_range=(0.3, 1.0),
        horizontal_flip=True,
        vertical_flip=True,
        fill_mode="nearest",
    )
    it = data_gen.flow(samples, batch_size=1)
    # generate samples and plot
    for i in range(no_of_samples):
        # generate batch of images
        batch = it.next()
        # convert to unsigned integers for viewing
        image = batch[0].astype("uint8")
        imgs.append(cv2.cvtColor(image, cv2.COLOR_RGB2GRAY))
        filename, label = get_filename_label(file_path)
        labels.append(get_one_hot_encoded_label(label))

    return imgs, labels


no_of_file = 100
seq = [i for i in range(len(onlyfiles))]
file_paths = [onlyfiles[random.choice(seq)] for _ in range(no_of_file)]
with cf.ProcessPoolExecutor() as executor:
    results = executor.map(augment_data, file_paths)

images = []
labels = []

for result in results:
    images.extend(result[0])
    labels.extend(result[1])

dataset_xc = np.append(dataset_xc, np.array(images), axis=0)
dataset_yc = np.append(dataset_yc, np.array(labels), axis=0)
```

### 1.0.22 Augmented Images

[17]:
```
plt.rcParams["figure.figsize"] = [16, 6]
fig, axs = plt.subplots(3, 3)
file_path = onlyfiles[55]
img = load_img(file_path)
axs[0, 0].set(title="Original Image")
axs[0, 0].imshow(img)
imgs = augment_data(file_path, 8)[0]
for i in range(1, 9):
    img = imgs[i - 1]
    axs[int(i / 3), i % 3].imshow(img)
```

```
plt.suptitle("Augmented Images")
plt.show()
del img, imgs
```




### 1.0.23 Reshape the dataset to be used with model

[18]:
```
print("dataset_yc:", dataset_yc.shape)
yc = dataset_yc.reshape(len(dataset_yc), -1)
print("yc:", yc.shape)
print("dataset_yc:", dataset_xc.shape)
xc = dataset_xc.reshape(len(dataset_xc), -1)
xc.shape

tensorflow.keras.utils.normalize(xc, axis=-1, order=2)
print("xc:", xc.shape)
```

```
dataset_yc: (3164, 3)
yc: (3164, 3)
dataset_yc: (3164, 256, 256)
xc: (3164, 65536)
```

### 1.0.24 Split dataset in test set and training sets

[19]:
```
train_xc, test_xc, train_yc, test_yc, = train_test_split(
    xc, yc, train_size=0.80, random_state=seed
)
print("Training Set length", len(train_xc))
```

```
print("Test Set length", len(test_xc))
```

```
Training Set length 2531
Test Set length 633
```

[20]:
```
train_xc.shape
```

[20]:
```
(2531, 65536)
```

### 1.0.25 Function plot model comparisons to tune hyperparameters

[21]:
```
def plot_comparison(histories, legends, title):
    fig, axs = plt.subplots(2, 2)
    for history in histories:
        rounds = len(history.history["loss"])
        ax1 = sns.lineplot(
            x=range(1, rounds + 1),
            y=history.history["loss"],
            legend="full",
            ax=axs[0, 0],
        )
        ax1.set(title="loss")
        ax1.legend(legends)
        ax2 = sns.lineplot(
            x=range(1, rounds + 1),
            y=history.history["val_loss"],
            legend="full",
            ax=axs[0, 1],
        )
        ax2.set(title="val_loss")
        ax2.legend(legends)
        ax3 = sns.lineplot(
            x=range(1, rounds + 1),
            y=history.history["accuracy"],
            legend="full",
            ax=axs[1, 0],
        )
        ax3.set(title="accuracy")
        ax3.legend(legends)
        ax4 = sns.lineplot(
            x=range(1, rounds + 1),
            y=history.history["val_accuracy"],
            legend="full",
            ax=axs[1, 1],
        )
        ax4.set(title="val_accuracy")
        ax4.legend(legends)
```

```
    plt.suptitle(title)
    plt.show()
```

### 1.0.26 Build the model for training

[22]:
```
input_shape = 65536

from tensorflow.keras import regularizers


def create_keras_model(
    optimizer=keras.optimizers.Adagrad, reg_param=None, learning_rate=0.01
):
    model = Sequential()
    model.add(Input(input_shape))
    # standardizes the inputs to a layer for each mini-batch.
    model.add(BatchNormalization(axis=1, name="bn0"))
    model.add(Activation("relu"))
    if reg_param is None:
        dense = Dense(100, activation="relu")
    else:
        dense = Dense(100, activation="relu", **reg_param)

    model.add(dense)
    model.add(Activation("relu"))
    model.add(Dense(3, activation="softmax"))

    opt = optimizer(learning_rate=learning_rate)
    model.compile(loss="categorical_crossentropy", optimizer=opt,␣
 ↪metrics=["accuracy"])

    return model
```

### 1.0.27 Compare model for different Batch Sizes

[23]:
```
histories = []
for batch_size in [30, 50, 70]:
    model = create_keras_model(keras.optimizers.Adagrad)
    print("\n\n", "*" * 50, "Batch Size:", batch_size, "*" * 50)
    model.summary()
    history = model.fit(
        x=train_xc,
        y=train_yc,
        batch_size=batch_size,
        epochs=20,
```

```
        validation_split=0.2,
        verbose=1,
        # steps_per_epoch=30,
    )
    histories.append(history)
    print(history.history)

plot_comparison(
    histories, ["30", "50", "70"], "Model comparison with Batch Sizes␣
 ↪Parameters",
)
```

```
 ************************************************** Batch Size: 30
**************************************************
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
activation (Activation)      (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 100)               6553700
_________________________________________________________________
activation_1 (Activation)    (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 303
=================================================================
Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072
_________________________________________________________________
Epoch 1/20
68/68 [==============================] - 1s 21ms/step - loss: 1.6879 - accuracy:
0.5499 - val_loss: 2.4378 - val_accuracy: 0.3077
Epoch 2/20
68/68 [==============================] - 1s 16ms/step - loss: 0.7698 - accuracy:
0.7050 - val_loss: 3.0425 - val_accuracy: 0.4714
Epoch 3/20
68/68 [==============================] - 1s 16ms/step - loss: 0.6023 - accuracy:
0.7584 - val_loss: 0.6700 - val_accuracy: 0.6943
Epoch 4/20
68/68 [==============================] - 1s 16ms/step - loss: 0.4814 - accuracy:
0.8127 - val_loss: 1.2178 - val_accuracy: 0.5424
Epoch 5/20
```

```
68/68 [==============================] - 1s 16ms/step - loss: 0.4328 - accuracy:
0.8325 - val_loss: 1.2746 - val_accuracy: 0.6331
Epoch 6/20
68/68 [==============================] - 1s 16ms/step - loss: 0.3856 - accuracy:
0.8434 - val_loss: 0.5221 - val_accuracy: 0.7830
Epoch 7/20
68/68 [==============================] - 1s 15ms/step - loss: 0.3438 - accuracy:
0.8725 - val_loss: 0.7889 - val_accuracy: 0.6963
Epoch 8/20
68/68 [==============================] - 1s 17ms/step - loss: 0.3079 - accuracy:
0.8903 - val_loss: 0.4895 - val_accuracy: 0.8107
Epoch 9/20
68/68 [==============================] - 1s 17ms/step - loss: 0.2807 - accuracy:
0.8992 - val_loss: 0.6437 - val_accuracy: 0.7574
Epoch 10/20
68/68 [==============================] - 1s 16ms/step - loss: 0.2421 - accuracy:
0.9096 - val_loss: 0.3782 - val_accuracy: 0.8422
Epoch 11/20
68/68 [==============================] - 1s 16ms/step - loss: 0.2270 - accuracy:
0.9214 - val_loss: 0.3908 - val_accuracy: 0.8540
Epoch 12/20
68/68 [==============================] - 1s 16ms/step - loss: 0.1861 - accuracy:
0.9353 - val_loss: 0.4998 - val_accuracy: 0.7673
Epoch 13/20
68/68 [==============================] - 1s 16ms/step - loss: 0.1832 - accuracy:
0.9402 - val_loss: 0.3818 - val_accuracy: 0.8619
Epoch 14/20
68/68 [==============================] - 1s 16ms/step - loss: 0.1750 - accuracy:
0.9368 - val_loss: 0.5178 - val_accuracy: 0.8185
Epoch 15/20
68/68 [==============================] - 1s 16ms/step - loss: 0.1548 - accuracy:
0.9466 - val_loss: 0.4644 - val_accuracy: 0.8323
Epoch 16/20
68/68 [==============================] - 1s 16ms/step - loss: 0.1606 - accuracy:
0.9447 - val_loss: 0.3564 - val_accuracy: 0.8738
Epoch 17/20
68/68 [==============================] - 1s 15ms/step - loss: 0.1656 - accuracy:
0.9476 - val_loss: 0.4643 - val_accuracy: 0.8363
Epoch 18/20
68/68 [==============================] - 1s 17ms/step - loss: 0.1122 - accuracy:
0.9694 - val_loss: 0.6787 - val_accuracy: 0.7811
Epoch 19/20
68/68 [==============================] - 1s 19ms/step - loss: 0.0993 - accuracy:
0.9738 - val_loss: 0.3843 - val_accuracy: 0.8600
Epoch 20/20
68/68 [==============================] - 1s 17ms/step - loss: 0.0994 - accuracy:
0.9748 - val_loss: 0.3797 - val_accuracy: 0.8540
{'loss': [1.6879299879074097, 0.7697572708129883, 0.6023337841033936,
```

0.48136842250823975, 0.43278786540031433, 0.3855997323989868,
0.3437800705432892, 0.30785199999809265, 0.28074800968170166,
0.24212342500686646, 0.22698983550071716, 0.18606193363666534,
0.1832144409418106, 0.1750481277704239, 0.15475930273532867,
0.16063569486141205, 0.16556298732757568, 0.11224064230918884,
0.09928147494792938, 0.09935958683490753], 'accuracy': [0.5499011874198914,
0.7050395011901855, 0.7583991885185242, 0.812747061252594, 0.8325098752975464,
0.8433794379234314, 0.8725296258926392, 0.8903161883354187, 0.8992094993591309,
0.9095849990844727, 0.9214426875114441, 0.9352766871452332, 0.9402173757553101,
0.9367588758468628, 0.9466403126716614, 0.9446640610694885, 0.9476284384727478,
0.9693675637245178, 0.9738142490386963, 0.9748023748397827], 'val_loss':
[2.4377973079681396, 3.0424835681915283, 0.6699713468551636, 1.217821717262268,
1.2746042013168335, 0.522071897983551, 0.788933277130127, 0.48951026797294617,
0.6437044739723206, 0.37816309928894043, 0.39079830050468445,
0.4997952878475189, 0.38179856538772583, 0.5177686214447021, 0.4643595516681671,
0.35635027289390564, 0.4642994999885559, 0.6787379384040833,
0.38431403040885925, 0.37970370054244995], 'val_accuracy': [0.3076923191547394,
0.4714003801345825, 0.6942800879478455, 0.5424063205718994, 0.6331360936164856,
0.7830374836921692, 0.6962524652481079, 0.8106508851051331, 0.7573964595794678,
0.8422091007232666, 0.8540433645248413, 0.7672584056854248, 0.8619329333305359,
0.8185404539108276, 0.8323471546173096, 0.8737672567367554, 0.8362919092178345,
0.7810651063919067, 0.8599605560302734, 0.8540433645248413]}

********** Batch Size: 50
**********
Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| bn0 (BatchNormalization) | (None, 65536) | 262144 |
| activation_2 (Activation) | (None, 65536) | 0 |
| dense_2 (Dense) | (None, 100) | 6553700 |
| activation_3 (Activation) | (None, 100) | 0 |
| dense_3 (Dense) | (None, 3) | 303 |

Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072

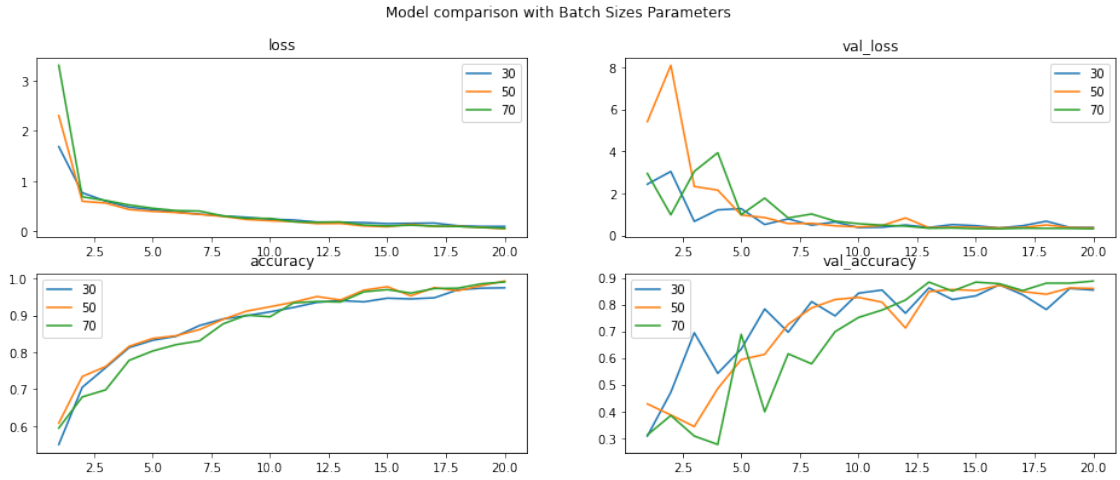
Epoch 1/20
41/41 [==============================] - 1s 32ms/step - loss: 2.3055 - accuracy:
0.6077 - val_loss: 5.4119 - val_accuracy: 0.4280
Epoch 2/20

```
41/41 [==============================] - 1s 23ms/step - loss: 0.5992 - accuracy:
0.7342 - val_loss: 8.0862 - val_accuracy: 0.3866
Epoch 3/20
41/41 [==============================] - 1s 23ms/step - loss: 0.5646 - accuracy:
0.7614 - val_loss: 2.3299 - val_accuracy: 0.3432
Epoch 4/20
41/41 [==============================] - 1s 23ms/step - loss: 0.4357 - accuracy:
0.8162 - val_loss: 2.1562 - val_accuracy: 0.4852
Epoch 5/20
41/41 [==============================] - 1s 23ms/step - loss: 0.3981 - accuracy:
0.8375 - val_loss: 0.9731 - val_accuracy: 0.5937
Epoch 6/20
41/41 [==============================] - 1s 23ms/step - loss: 0.3760 - accuracy:
0.8449 - val_loss: 0.8508 - val_accuracy: 0.6134
Epoch 7/20
41/41 [==============================] - 1s 23ms/step - loss: 0.3387 - accuracy:
0.8617 - val_loss: 0.5672 - val_accuracy: 0.7258
Epoch 8/20
41/41 [==============================] - 1s 24ms/step - loss: 0.2978 - accuracy:
0.8893 - val_loss: 0.5768 - val_accuracy: 0.7870
Epoch 9/20
41/41 [==============================] - 1s 24ms/step - loss: 0.2376 - accuracy:
0.9116 - val_loss: 0.4589 - val_accuracy: 0.8185
Epoch 10/20
41/41 [==============================] - 1s 24ms/step - loss: 0.2120 - accuracy:
0.9234 - val_loss: 0.4070 - val_accuracy: 0.8264
Epoch 11/20
41/41 [==============================] - 1s 24ms/step - loss: 0.1943 - accuracy:
0.9358 - val_loss: 0.4748 - val_accuracy: 0.8087
Epoch 12/20
41/41 [==============================] - 1s 24ms/step - loss: 0.1553 - accuracy:
0.9511 - val_loss: 0.8313 - val_accuracy: 0.7120
Epoch 13/20
41/41 [==============================] - 1s 24ms/step - loss: 0.1582 - accuracy:
0.9417 - val_loss: 0.3707 - val_accuracy: 0.8481
Epoch 14/20
41/41 [==============================] - 1s 24ms/step - loss: 0.1084 - accuracy:
0.9684 - val_loss: 0.3970 - val_accuracy: 0.8560
Epoch 15/20
41/41 [==============================] - 1s 24ms/step - loss: 0.0922 - accuracy:
0.9778 - val_loss: 0.3860 - val_accuracy: 0.8521
Epoch 16/20
41/41 [==============================] - 1s 24ms/step - loss: 0.1405 - accuracy:
0.9531 - val_loss: 0.3607 - val_accuracy: 0.8718
Epoch 17/20
41/41 [==============================] - 1s 24ms/step - loss: 0.0993 - accuracy:
0.9753 - val_loss: 0.3791 - val_accuracy: 0.8481
Epoch 18/20
```

```
41/41 [==============================] - 1s 24ms/step - loss: 0.1010 - accuracy:
0.9669 - val_loss: 0.5003 - val_accuracy: 0.8383
Epoch 19/20
41/41 [==============================] - 1s 24ms/step - loss: 0.0771 - accuracy:
0.9807 - val_loss: 0.3772 - val_accuracy: 0.8619
Epoch 20/20
41/41 [==============================] - 1s 25ms/step - loss: 0.0520 - accuracy:
0.9931 - val_loss: 0.3579 - val_accuracy: 0.8600
{'loss': [2.3055036067962646, 0.5992028713226318, 0.5645537972450256,
0.43565747141838074, 0.3981219530105591, 0.37602922320365906,
0.3387133479118347, 0.29776087403297424, 0.23762400448322296,
0.21197815239429474, 0.19428306818008423, 0.1553177833557129,
0.15815412998199463, 0.10840919613838196, 0.092216856777668,
0.14052586257457733, 0.0992870032787323, 0.10103018581867218,
0.07709486782550812, 0.05195072293281555], 'accuracy': [0.6077075004577637,
0.7341897487640381, 0.7613636255264282, 0.8162055611610413, 0.8374505639076233,
0.8448616862297058, 0.8616600632667542, 0.8893280625343323, 0.9115612506866455,
0.9234189987182617, 0.9357707500457764, 0.9510869383811951, 0.9416996240615845,
0.9683794379234314, 0.9777668118476868, 0.9530632495880127, 0.9752964377403259,
0.9668972492218018, 0.9807312488555908, 0.9930830001831055], 'val_loss':
[5.411865234375, 8.086241722106934, 2.329878568649292, 2.1561782360076904,
0.9731194376945496, 0.8508071899414062, 0.5671729445457458, 0.5767754316329956,
0.45888516306877136, 0.4069643020629883, 0.47481632232666016,
0.8312656283378601, 0.37071681022644043, 0.3970363736152649,
0.38602960109710693, 0.36066004633903503, 0.37907299399375916,
0.5003486275672913, 0.37722912430763245, 0.35790130496025085], 'val_accuracy':
[0.42800790071487427, 0.3865877687931061, 0.34319525957107544,
0.48520711064338684, 0.5936883687973022, 0.6134122014045715, 0.7258382439613342,
0.7869822382926941, 0.8185404539108276, 0.8264299631118774, 0.8086785078048706,
0.7120315432548523, 0.848126232624054, 0.8560158014297485, 0.8520709872245789,
0.8717948794364929, 0.848126232624054, 0.8382642865180969, 0.8619329333305359,
0.8599605560302734]}


 ************************************************ Batch Size: 70
************************************************
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
activation_4 (Activation)    (None, 65536)             0
_________________________________________________________________
dense_4 (Dense)              (None, 100)               6553700
_________________________________________________________________
activation_5 (Activation)    (None, 100)               0
_________________________________________________________________
```

```
dense_5 (Dense)              (None, 3)                 303
=================================================================
Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072
_________________________________________________________________
Epoch 1/20
29/29 [==============================] - 1s 42ms/step - loss: 3.3064 - accuracy:
0.5944 - val_loss: 2.9482 - val_accuracy: 0.3116
Epoch 2/20
29/29 [==============================] - 1s 36ms/step - loss: 0.6881 - accuracy:
0.6789 - val_loss: 0.9809 - val_accuracy: 0.3846
Epoch 3/20
29/29 [==============================] - 1s 33ms/step - loss: 0.6156 - accuracy:
0.6981 - val_loss: 3.0455 - val_accuracy: 0.3077
Epoch 4/20
29/29 [==============================] - 1s 33ms/step - loss: 0.5262 - accuracy:
0.7782 - val_loss: 3.9334 - val_accuracy: 0.2761
Epoch 5/20
29/29 [==============================] - 1s 32ms/step - loss: 0.4605 - accuracy:
0.8034 - val_loss: 0.9882 - val_accuracy: 0.6884
Epoch 6/20
29/29 [==============================] - 1s 32ms/step - loss: 0.4148 - accuracy:
0.8207 - val_loss: 1.7755 - val_accuracy: 0.3984
Epoch 7/20
29/29 [==============================] - 1s 33ms/step - loss: 0.4049 - accuracy:
0.8310 - val_loss: 0.8334 - val_accuracy: 0.6154
Epoch 8/20
29/29 [==============================] - 1s 33ms/step - loss: 0.3077 - accuracy:
0.8770 - val_loss: 1.0210 - val_accuracy: 0.5779
Epoch 9/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2603 - accuracy:
0.9007 - val_loss: 0.6819 - val_accuracy: 0.6982
Epoch 10/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2578 - accuracy:
0.8962 - val_loss: 0.5605 - val_accuracy: 0.7515
Epoch 11/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1918 - accuracy:
0.9338 - val_loss: 0.4901 - val_accuracy: 0.7791
Epoch 12/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1814 - accuracy:
0.9377 - val_loss: 0.4454 - val_accuracy: 0.8166
Epoch 13/20
29/29 [==============================] - 1s 34ms/step - loss: 0.1892 - accuracy:
0.9363 - val_loss: 0.3445 - val_accuracy: 0.8836
Epoch 14/20
29/29 [==============================] - 1s 34ms/step - loss: 0.1275 - accuracy:
0.9639 - val_loss: 0.3576 - val_accuracy: 0.8501
```

```
Epoch 15/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1088 - accuracy:
0.9699 - val_loss: 0.3241 - val_accuracy: 0.8836
Epoch 16/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1243 - accuracy:
0.9605 - val_loss: 0.3190 - val_accuracy: 0.8777
Epoch 17/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1048 - accuracy:
0.9728 - val_loss: 0.3452 - val_accuracy: 0.8521
Epoch 18/20
29/29 [==============================] - 1s 33ms/step - loss: 0.0992 - accuracy:
0.9738 - val_loss: 0.3413 - val_accuracy: 0.8797
Epoch 19/20
29/29 [==============================] - 1s 33ms/step - loss: 0.0777 - accuracy:
0.9857 - val_loss: 0.3344 - val_accuracy: 0.8797
Epoch 20/20
29/29 [==============================] - 1s 33ms/step - loss: 0.0620 - accuracy:
0.9906 - val_loss: 0.3191 - val_accuracy: 0.8876
{'loss': [3.306443691253662, 0.6881363987922668, 0.6155716180801392,
0.526211678981781, 0.46051058173179626, 0.4147583544254303, 0.4049418866634369,
0.307740718126297, 0.2603442370891571, 0.2577618658542633, 0.19177140295505524,
0.18141628801822662, 0.18921495974063873, 0.1275235116481781,
0.10875622183084488, 0.1242820993065834, 0.10480232536792755,
0.09923425316810608, 0.07766274362802505, 0.062045544385910034], 'accuracy':
[0.5943675637245178, 0.6788537502288818, 0.698122501373291, 0.7781620621681213,
0.8033596873283386, 0.820652186870575, 0.8310276865959167, 0.8769763112068176,
0.9006916880607605, 0.8962450623512268, 0.9337944388389587, 0.937747061252594,
0.9362648129463196, 0.9639328122138977, 0.9698616862297058, 0.9604743123054504,
0.9728260636329651, 0.9738142490386963, 0.9856719374656677, 0.9906126260757446],
'val_loss': [2.9481518268585205, 0.9809068441390991, 3.0454742908477783,
3.9333629608154297, 0.9882104396820068, 1.7755078077316284, 0.8334043025970459,
1.0210233926773071, 0.6819071173667908, 0.5605086088180542, 0.4901202321052551,
0.44539931416511536, 0.34454241394996643, 0.3576192557811737,
0.32411205768585205, 0.3189554512500763, 0.34516671299934387,
0.3413315415382385, 0.3344259560108185, 0.3190664052963257], 'val_accuracy':
[0.3116370737552643, 0.38461539149284363, 0.3076923191547394,
0.2761341333389282, 0.6883628964424133, 0.39842209219932556, 0.6153846383094788,
0.5779092907905579, 0.6982248425483704, 0.7514792680740356, 0.7790927290916443,
0.8165680766105652, 0.8836292028427124, 0.8500986099243164, 0.8836292028427124,
0.8777120113372803, 0.8520709872245789, 0.8796843886375427, 0.8796843886375427,
0.8875739574432373]}
```




### 1.0.28 Compare kernel regularizers

[24]:
```
histories = []
batch_size = 70
kr = {
    "kernel_regularizer": regularizers.l1_l2(l1=1e-5, l2=1e-4),
}
br = {
    "bias_regularizer": regularizers.l2(1e-4),
}
ar = {
    "activity_regularizer": regularizers.l2(1e-5),
}
regulairization_param = [kr, br, ar]

for reg_param in regulairization_param:
    model = create_keras_model(keras.optimizers.Adagrad, reg_param)
    print("\n\n", "*" * 50, "Regularizer:", list(reg_param.keys())[0], "*" * 50)
    model.summary()
    history = model.fit(
        x=train_xc,
        y=train_yc,
        batch_size=batch_size,
        epochs=20,
        validation_split=0.2,
        verbose=1,
        # steps_per_epoch=20,
    )
    histories.append(history)
    print(history.history)
```

```
plot_comparison(
    histories,
    [
        list(regulairization_param[0].keys())[0],
        list(regulairization_param[1].keys())[0],
        list(regulairization_param[2].keys())[0],
    ],
    "Model comparison with Regularization Parameters",
)
```

```
 ************************************************ Regularizer:
kernel_regularizer ************************************************
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
activation_6 (Activation)    (None, 65536)             0
_________________________________________________________________
dense_6 (Dense)              (None, 100)               6553700
_________________________________________________________________
activation_7 (Activation)    (None, 100)               0
_________________________________________________________________
dense_7 (Dense)              (None, 3)                 303
=================================================================
Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072
_________________________________________________________________
Epoch 1/20
29/29 [==============================] - 1s 42ms/step - loss: 3.6961 - accuracy:
0.5879 - val_loss: 6.7949 - val_accuracy: 0.4300
Epoch 2/20
29/29 [==============================] - 1s 34ms/step - loss: 0.9694 - accuracy:
0.7184 - val_loss: 4.2686 - val_accuracy: 0.2604
Epoch 3/20
29/29 [==============================] - 1s 33ms/step - loss: 0.9219 - accuracy:
0.7510 - val_loss: 4.3664 - val_accuracy: 0.4497
Epoch 4/20
29/29 [==============================] - 1s 33ms/step - loss: 0.7878 - accuracy:
0.8063 - val_loss: 4.0500 - val_accuracy: 0.4556
Epoch 5/20
29/29 [==============================] - 1s 32ms/step - loss: 0.7814 - accuracy:
```

```
0.8103 - val_loss: 2.8387 - val_accuracy: 0.4320
Epoch 6/20
29/29 [==============================] - 1s 34ms/step - loss: 0.7015 - accuracy:
0.8429 - val_loss: 1.9284 - val_accuracy: 0.4418
Epoch 7/20
29/29 [==============================] - 1s 33ms/step - loss: 0.6562 - accuracy:
0.8671 - val_loss: 1.3169 - val_accuracy: 0.5976
Epoch 8/20
29/29 [==============================] - 1s 35ms/step - loss: 0.6252 - accuracy:
0.8814 - val_loss: 1.4284 - val_accuracy: 0.5562
Epoch 9/20
29/29 [==============================] - 1s 36ms/step - loss: 0.5834 - accuracy:
0.9012 - val_loss: 0.9356 - val_accuracy: 0.7456
Epoch 10/20
29/29 [==============================] - 1s 34ms/step - loss: 0.6254 - accuracy:
0.8829 - val_loss: 0.8469 - val_accuracy: 0.7692
Epoch 11/20
29/29 [==============================] - 1s 33ms/step - loss: 0.5176 - accuracy:
0.9363 - val_loss: 0.7021 - val_accuracy: 0.8501
Epoch 12/20
29/29 [==============================] - 1s 35ms/step - loss: 0.4851 - accuracy:
0.9545 - val_loss: 0.8396 - val_accuracy: 0.7909
Epoch 13/20
29/29 [==============================] - 1s 35ms/step - loss: 0.5088 - accuracy:
0.9373 - val_loss: 0.6515 - val_accuracy: 0.8777
Epoch 14/20
29/29 [==============================] - 1s 34ms/step - loss: 0.4584 - accuracy:
0.9629 - val_loss: 0.6912 - val_accuracy: 0.8540
Epoch 15/20
29/29 [==============================] - 1s 34ms/step - loss: 0.4340 - accuracy:
0.9728 - val_loss: 0.6818 - val_accuracy: 0.8698
Epoch 16/20
29/29 [==============================] - 1s 34ms/step - loss: 0.4398 - accuracy:
0.9718 - val_loss: 0.6575 - val_accuracy: 0.8738
Epoch 17/20
29/29 [==============================] - 1s 35ms/step - loss: 0.4354 - accuracy:
0.9718 - val_loss: 0.6647 - val_accuracy: 0.8718
Epoch 18/20
29/29 [==============================] - 1s 34ms/step - loss: 0.4221 - accuracy:
0.9723 - val_loss: 0.6434 - val_accuracy: 0.8876
Epoch 19/20
29/29 [==============================] - 1s 34ms/step - loss: 0.4175 - accuracy:
0.9778 - val_loss: 0.6477 - val_accuracy: 0.8757
Epoch 20/20
29/29 [==============================] - 1s 35ms/step - loss: 0.3967 - accuracy:
0.9876 - val_loss: 0.6731 - val_accuracy: 0.8679
{'loss': [3.6960511207580566, 0.9693978428840637, 0.9218827486038208,
0.7877508401870728, 0.781352698802948, 0.7014671564102173, 0.6562280654907227,
```

0.6252172589302063, 0.5833746194839478, 0.6253852248191833, 0.5175676941871643,
0.4850778579711914, 0.5088433027267456, 0.45836538076400757,
0.43404316902160645, 0.4398379921913147, 0.43536603450775146,
0.4221043586730957, 0.41754859685897827, 0.39670896530151367], 'accuracy':
[0.5879446864128113, 0.7183794379234314, 0.7509881258010864, 0.8063241243362427,
0.8102766871452332, 0.8428853750228882, 0.867094874382019, 0.8814229369163513,
0.9011857509613037, 0.882905125617981, 0.9362648129463196, 0.9545454382896423,
0.937252938747406, 0.9629446864128113, 0.9728260636329651, 0.9718379378318787,
0.9718379378318787, 0.9723320007324219, 0.9777668118476868, 0.9876482486724854],
'val_loss': [6.794933319091797, 4.268645286560059, 4.366387844085693,
4.050026893615723, 2.8386998176574707, 1.92840576171875, 1.3169069290161133,
1.4283684492111206, 0.9356033802032471, 0.8468610048294067, 0.7020574808120728,
0.8395591974258423, 0.6514846682548523, 0.691218912601471, 0.6817587614059448,
0.6575461030006409, 0.6646512150764465, 0.6433563828468323, 0.6476535201072693,
0.6730836033821106], 'val_accuracy': [0.4299802780151367, 0.26035502552986145,
0.4497041404247284, 0.45562130212783813, 0.43195265531539917,
0.4418146014213562, 0.5976331233978271, 0.5562130212783813, 0.7455621361732483,
0.7692307829856873, 0.8500986099243164, 0.790926992893219, 0.8777120113372803,
0.8540433645248413, 0.8698225021362305, 0.8737672567367554, 0.8717948794364929,
0.8875739574432373, 0.8757396340370178, 0.867850124835968]}

********** Regularizer:
bias_regularizer **********

Model: "sequential_4"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| bn0 (BatchNormalization) | (None, 65536) | 262144 |
| activation_8 (Activation) | (None, 65536) | 0 |
| dense_8 (Dense) | (None, 100) | 6553700 |
| activation_9 (Activation) | (None, 100) | 0 |
| dense_9 (Dense) | (None, 3) | 303 |

Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072

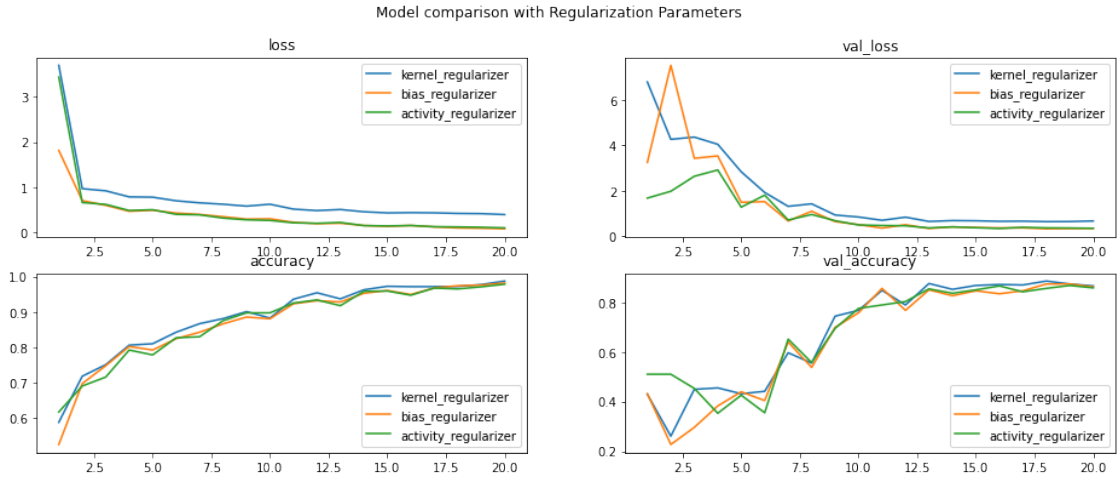
Epoch 1/20
29/29 [==============================] - 1s 43ms/step - loss: 1.8153 - accuracy:
0.5252 - val_loss: 3.2521 - val_accuracy: 0.4320
Epoch 2/20
29/29 [==============================] - 1s 33ms/step - loss: 0.7054 - accuracy:
0.6976 - val_loss: 7.5261 - val_accuracy: 0.2268

```
Epoch 3/20
29/29 [==============================] - 1s 34ms/step - loss: 0.6008 - accuracy:
0.7480 - val_loss: 3.4317 - val_accuracy: 0.2959
Epoch 4/20
29/29 [==============================] - 1s 33ms/step - loss: 0.4699 - accuracy:
0.8024 - val_loss: 3.5389 - val_accuracy: 0.3826
Epoch 5/20
29/29 [==============================] - 1s 45ms/step - loss: 0.4920 - accuracy:
0.7925 - val_loss: 1.4968 - val_accuracy: 0.4398
Epoch 6/20
29/29 [==============================] - 1s 33ms/step - loss: 0.4303 - accuracy:
0.8246 - val_loss: 1.5252 - val_accuracy: 0.4043
Epoch 7/20
29/29 [==============================] - 1s 32ms/step - loss: 0.3998 - accuracy:
0.8429 - val_loss: 0.6725 - val_accuracy: 0.6410
Epoch 8/20
29/29 [==============================] - 1s 33ms/step - loss: 0.3498 - accuracy:
0.8666 - val_loss: 1.1004 - val_accuracy: 0.5385
Epoch 9/20
29/29 [==============================] - 1s 32ms/step - loss: 0.2963 - accuracy:
0.8859 - val_loss: 0.6457 - val_accuracy: 0.7002
Epoch 10/20
29/29 [==============================] - 1s 33ms/step - loss: 0.3036 - accuracy:
0.8809 - val_loss: 0.5113 - val_accuracy: 0.7594
Epoch 11/20
29/29 [==============================] - 1s 32ms/step - loss: 0.2261 - accuracy:
0.9234 - val_loss: 0.3585 - val_accuracy: 0.8580
Epoch 12/20
29/29 [==============================] - 1s 37ms/step - loss: 0.1969 - accuracy:
0.9323 - val_loss: 0.5081 - val_accuracy: 0.7692
Epoch 13/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2054 - accuracy:
0.9284 - val_loss: 0.3402 - val_accuracy: 0.8521
Epoch 14/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1552 - accuracy:
0.9531 - val_loss: 0.4123 - val_accuracy: 0.8284
Epoch 15/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1367 - accuracy:
0.9610 - val_loss: 0.3842 - val_accuracy: 0.8481
Epoch 16/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1561 - accuracy:
0.9496 - val_loss: 0.3677 - val_accuracy: 0.8363
Epoch 17/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1232 - accuracy:
0.9684 - val_loss: 0.3689 - val_accuracy: 0.8481
Epoch 18/20
29/29 [==============================] - 1s 36ms/step - loss: 0.1012 - accuracy:
0.9743 - val_loss: 0.3293 - val_accuracy: 0.8757
```

```
Epoch 19/20
29/29 [==============================] - 1s 34ms/step - loss: 0.0935 - accuracy:
0.9763 - val_loss: 0.3391 - val_accuracy: 0.8757
Epoch 20/20
29/29 [==============================] - 1s 32ms/step - loss: 0.0828 - accuracy:
0.9822 - val_loss: 0.3472 - val_accuracy: 0.8639
{'loss': [1.815311312675476, 0.7054116129875183, 0.6008328199386597,
0.46994441747665405, 0.49196764826774597, 0.43030768632888794,
0.3997974991798401, 0.3497955799102783, 0.29625454545021057, 0.3036438226699829,
0.2261025458574295, 0.196903795003891, 0.20543675124645233, 0.15520833432674408,
0.13674573600292206, 0.15613244473934174, 0.12317534536123276,
0.10117154568433762, 0.0935470312833786, 0.08275298774242401], 'accuracy':
[0.5251976251602173, 0.6976284384727478, 0.7480236887931824, 0.8023715615272522,
0.7924901247024536, 0.8246047496795654, 0.8428853750228882, 0.8666008114814758,
0.885869562625885, 0.8809288740158081, 0.9234189987182617, 0.9323122501373291,
0.9283596873283386, 0.9530632495880127, 0.9609683752059937, 0.9496047496795654,
0.9683794379234314, 0.9743083119392395, 0.9762845635414124, 0.9822134375572205],
'val_loss': [3.252082586288452, 7.526129722595215, 3.431718349456787,
3.538874864578247, 1.4967987537384033, 1.5251564979553223, 0.6724798083305359,
1.1003749370574951, 0.6457021832466125, 0.511278510093689, 0.3584855794906616,
0.5080881714820862, 0.34016379714012146, 0.41227421164512634,
0.38420698046684265, 0.36765098571777344, 0.36886680126190186,
0.32930827140808105, 0.3391377925872803, 0.3472388982772827], 'val_accuracy':
[0.43195265531539917, 0.22682446241378784, 0.2958579957485199,
0.3826429843902588, 0.43984219431877136, 0.4043392539024353, 0.6410256624221802,
0.5384615659713745, 0.7001972198486328, 0.7593688368797302, 0.857988178730011,
0.7692307829856873, 0.8520709872245789, 0.8284023404121399, 0.848126232624054,
0.8362919092178345, 0.848126232624054, 0.8757396340370178, 0.8757396340370178,
0.8639053106307983]}


 ********************************************** Regularizer:
activity_regularizer ************************************************
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
activation_10 (Activation)   (None, 65536)             0
_________________________________________________________________
dense_10 (Dense)             (None, 100)               6553700
_________________________________________________________________
activation_11 (Activation)   (None, 100)               0
_________________________________________________________________
dense_11 (Dense)             (None, 3)                 303
=================================================================
Total params: 6,816,147
```

```
Trainable params: 6,685,075
Non-trainable params: 131,072
_________________________________________________________________
Epoch 1/20
29/29 [==============================] - 3s 110ms/step - loss: 3.4365 -
accuracy: 0.6171 - val_loss: 1.6805 - val_accuracy: 0.5108
Epoch 2/20
29/29 [==============================] - 1s 33ms/step - loss: 0.6621 - accuracy:
0.6907 - val_loss: 1.9803 - val_accuracy: 0.5108
Epoch 3/20
29/29 [==============================] - 1s 33ms/step - loss: 0.6215 - accuracy:
0.7159 - val_loss: 2.6408 - val_accuracy: 0.4536
Epoch 4/20
29/29 [==============================] - 1s 34ms/step - loss: 0.4871 - accuracy:
0.7925 - val_loss: 2.9189 - val_accuracy: 0.3531
Epoch 5/20
29/29 [==============================] - 1s 34ms/step - loss: 0.5040 - accuracy:
0.7787 - val_loss: 1.2813 - val_accuracy: 0.4260
Epoch 6/20
29/29 [==============================] - 1s 33ms/step - loss: 0.4016 - accuracy:
0.8271 - val_loss: 1.8105 - val_accuracy: 0.3550
Epoch 7/20
29/29 [==============================] - 1s 32ms/step - loss: 0.3912 - accuracy:
0.8300 - val_loss: 0.7145 - val_accuracy: 0.6529
Epoch 8/20
29/29 [==============================] - 1s 33ms/step - loss: 0.3186 - accuracy:
0.8755 - val_loss: 0.9575 - val_accuracy: 0.5582
Epoch 9/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2809 - accuracy:
0.8977 - val_loss: 0.6857 - val_accuracy: 0.6963
Epoch 10/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2672 - accuracy:
0.8977 - val_loss: 0.4976 - val_accuracy: 0.7771
Epoch 11/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2145 - accuracy:
0.9254 - val_loss: 0.4740 - val_accuracy: 0.7909
Epoch 12/20
29/29 [==============================] - 1s 34ms/step - loss: 0.2037 - accuracy:
0.9343 - val_loss: 0.4608 - val_accuracy: 0.8047
Epoch 13/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2224 - accuracy:
0.9180 - val_loss: 0.3670 - val_accuracy: 0.8560
Epoch 14/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1525 - accuracy:
0.9585 - val_loss: 0.4141 - val_accuracy: 0.8383
Epoch 15/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1450 - accuracy:
0.9595 - val_loss: 0.3755 - val_accuracy: 0.8521
```

```
Epoch 16/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1549 - accuracy:
0.9476 - val_loss: 0.3417 - val_accuracy: 0.8679
Epoch 17/20
29/29 [==============================] - 1s 34ms/step - loss: 0.1287 - accuracy:
0.9679 - val_loss: 0.3933 - val_accuracy: 0.8442
Epoch 18/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1238 - accuracy:
0.9654 - val_loss: 0.3693 - val_accuracy: 0.8580
Epoch 19/20
29/29 [==============================] - 1s 35ms/step - loss: 0.1163 - accuracy:
0.9713 - val_loss: 0.3627 - val_accuracy: 0.8698
Epoch 20/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1025 - accuracy:
0.9788 - val_loss: 0.3509 - val_accuracy: 0.8600
{'loss': [3.4365005493164062, 0.6620755791664124, 0.6214979887008667,
0.4871453046798706, 0.5040283799171448, 0.4015588164329529, 0.39123833179473877,
0.3186309039592743, 0.28094276785850525, 0.26720085740089417,
0.21448135375976562, 0.20369958877563477, 0.22242099046707153,
0.15251362323760986, 0.1449998915195465, 0.1549099236726761,
0.12873464822769165, 0.12376994639635086, 0.11626488715410233,
0.10250454396009445], 'accuracy': [0.617094874382019, 0.6907114386558533,
0.7159090638160706, 0.7924901247024536, 0.7786561250686646, 0.8270751237869263,
0.8300395011901855, 0.8754940629005432, 0.8977272510528564, 0.8977272510528564,
0.9253952503204346, 0.9342885613441467, 0.9179841876029968, 0.9584980010986328,
0.959486186504364, 0.9476284384727478, 0.9678853750228882, 0.9654150009155273,
0.9713438749313354, 0.9787549376487732], 'val_loss': [1.6804927587509155,
1.9803476333618164, 2.640779733657837, 2.918889045715332, 1.2812985181808472,
1.8105486631393433, 0.7145006656646729, 0.9575456976890564, 0.685693621635437,
0.4976162910461426, 0.4739968180656433, 0.4607771337032318, 0.367041677236557,
0.4141138792037964, 0.37545156478881836, 0.34174802899360657,
0.3933255970478058, 0.3693270981311798, 0.36267781257629395,
0.3509049117565155], 'val_accuracy': [0.5108481049537659, 0.5108481049537659,
0.4536489248275757, 0.35305720567703247, 0.42603549361228943,
0.3550295829772949, 0.6528599858283997, 0.5581853985786438, 0.6962524652481079,
0.7771202921867371, 0.790926992893219, 0.8047337532043457, 0.8560158014297485,
0.8382642865180969, 0.8520709872245789, 0.867850124835968, 0.844181478023529,
0.857988178730011, 0.8698225021362305, 0.8599605560302734]}
```




### 1.0.29 Compare different optimizers

[25]:
```
histories_opt = []
for optimizer in [
    keras.optimizers.Adam,
    keras.optimizers.SGD,
    keras.optimizers.RMSprop,
    keras.optimizers.Adagrad,
    keras.optimizers.Adamax,
]:
    print("\n\n", "*" * 50, "Optimizer:", optimizer, "*" * 50)
    model = create_keras_model(optimizer, br)
    model.summary()
    history = model.fit(
        x=train_xc,
        y=train_yc,
        batch_size=70,
        epochs=20,
        validation_split=0.2,
        verbose=1,
    )
    print(history.history)
    histories_opt.append(history)
plot_comparison(
    histories_opt,
    ["adam", "sgd", "rmsprop", "adagrad", "adamax"],
    "Model comparison with Optimizers",
)
```

```
 ********************************************** Optimizer: <class
'tensorflow.python.keras.optimizer_v2.adam.Adam'>
**********************************************
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
activation_12 (Activation)   (None, 65536)             0
_________________________________________________________________
dense_12 (Dense)             (None, 100)               6553700
_________________________________________________________________
activation_13 (Activation)   (None, 100)               0
_________________________________________________________________
dense_13 (Dense)             (None, 3)                 303
=================================================================
Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072
_________________________________________________________________
Epoch 1/20
29/29 [==============================] - 1s 39ms/step - loss: 59.6630 -
accuracy: 0.6087 - val_loss: 784.9626 - val_accuracy: 0.2288
Epoch 2/20
29/29 [==============================] - 1s 30ms/step - loss: 3.9148 - accuracy:
0.7737 - val_loss: 53.8867 - val_accuracy: 0.3826
Epoch 3/20
29/29 [==============================] - 1s 30ms/step - loss: 1.1517 - accuracy:
0.8019 - val_loss: 14.0926 - val_accuracy: 0.3314
Epoch 4/20
29/29 [==============================] - 1s 31ms/step - loss: 0.9035 - accuracy:
0.8276 - val_loss: 39.2053 - val_accuracy: 0.3905
Epoch 5/20
29/29 [==============================] - 1s 31ms/step - loss: 0.7492 - accuracy:
0.8498 - val_loss: 13.5746 - val_accuracy: 0.4438
Epoch 6/20
29/29 [==============================] - 1s 30ms/step - loss: 0.2583 - accuracy:
0.9205 - val_loss: 11.6956 - val_accuracy: 0.4280
Epoch 7/20
29/29 [==============================] - 1s 30ms/step - loss: 0.1695 - accuracy:
0.9377 - val_loss: 3.2217 - val_accuracy: 0.6509
Epoch 8/20
29/29 [==============================] - 1s 31ms/step - loss: 0.1493 - accuracy:
0.9516 - val_loss: 4.2980 - val_accuracy: 0.6134
Epoch 9/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1659 - accuracy:
0.9466 - val_loss: 2.8095 - val_accuracy: 0.7337
```

```
Epoch 10/20
29/29 [==============================] - 1s 32ms/step - loss: 0.0945 - accuracy:
0.9605 - val_loss: 1.7493 - val_accuracy: 0.7495
Epoch 11/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1021 - accuracy:
0.9625 - val_loss: 1.3835 - val_accuracy: 0.8284
Epoch 12/20
29/29 [==============================] - 1s 34ms/step - loss: 0.1216 - accuracy:
0.9590 - val_loss: 1.5291 - val_accuracy: 0.7732
Epoch 13/20
29/29 [==============================] - 1s 34ms/step - loss: 0.0784 - accuracy:
0.9718 - val_loss: 1.7280 - val_accuracy: 0.7771
Epoch 14/20
29/29 [==============================] - 1s 33ms/step - loss: 0.0868 - accuracy:
0.9704 - val_loss: 1.2037 - val_accuracy: 0.8363
Epoch 15/20
29/29 [==============================] - 1s 33ms/step - loss: 0.0592 - accuracy:
0.9807 - val_loss: 1.3394 - val_accuracy: 0.8304
Epoch 16/20
29/29 [==============================] - 1s 33ms/step - loss: 0.0701 - accuracy:
0.9797 - val_loss: 1.1481 - val_accuracy: 0.8343
Epoch 17/20
29/29 [==============================] - 1s 33ms/step - loss: 0.0657 - accuracy:
0.9768 - val_loss: 1.0828 - val_accuracy: 0.8343
Epoch 18/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1346 - accuracy:
0.9684 - val_loss: 1.9012 - val_accuracy: 0.7771
Epoch 19/20
29/29 [==============================] - 1s 34ms/step - loss: 0.2944 - accuracy:
0.9130 - val_loss: 0.6266 - val_accuracy: 0.8205
Epoch 20/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2585 - accuracy:
0.8992 - val_loss: 0.7971 - val_accuracy: 0.8481
{'loss': [59.66301727294922, 3.914808511734009, 1.1517400741577148,
0.9035390615463257, 0.7492380738258362, 0.2583271265029907, 0.16945378482341766,
0.14928944408893585, 0.16592465341091156, 0.09453026205301285,
0.10208473354578018, 0.1215764507651329, 0.07837609201669693,
0.08684707432985306, 0.05920888110995293, 0.07009638845920563,
0.06568796187639236, 0.13462166488170624, 0.29442209005355835,
0.2585158050060272], 'accuracy': [0.6086956262588501, 0.7737154364585876,
0.801877498626709, 0.8275691866874695, 0.8498023748397827, 0.9204545617103577,
0.937747061252594, 0.9515810012817383, 0.9466403126716614, 0.9604743123054504,
0.9624505639076233, 0.9589921236038208, 0.9718379378318787, 0.970355749130249,
0.9807312488555908, 0.9797430634498596, 0.9767786264419556, 0.9683794379234314,
0.9130434989929199, 0.8992094993591309], 'val_loss': [784.9625854492188,
53.88671112060547, 14.092604637145996, 39.205345153808594, 13.574604988098145,
11.695592880249023, 3.221717357635498, 4.298027992248535, 2.8094770908355713,
1.7493070363998413, 1.383484959602356, 1.5291205644607544, 1.7279775142669678,
```

```
1.2036569118499756, 1.3393675088882446, 1.1480876207351685, 1.0827752351760864,
1.9012229442596436, 0.6265931129455566, 0.7970589399337769], 'val_accuracy':
[0.2287968397140503, 0.3826429843902588, 0.33136093616485596,
0.39053255319595337, 0.44378697872161865, 0.42800790071487427,
0.6508875489234924, 0.6134122014045715, 0.7337278127670288, 0.7495068907737732,
0.8284023404121399, 0.7731755375862122, 0.7771202921867371, 0.8362919092178345,
0.8303747773170471, 0.834319531917572, 0.834319531917572, 0.7771202921867371,
0.8205128312110901, 0.848126232624054]}


 ************************************************ Optimizer: <class
'tensorflow.python.keras.optimizer_v2.gradient_descent.SGD'>
************************************************
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
activation_14 (Activation)   (None, 65536)             0
_________________________________________________________________
dense_14 (Dense)             (None, 100)               6553700
_________________________________________________________________
activation_15 (Activation)   (None, 100)               0
_________________________________________________________________
dense_15 (Dense)             (None, 3)                 303
=================================================================
Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072
_________________________________________________________________
Epoch 1/20
29/29 [==============================] - 1s 43ms/step - loss: 1.5737 - accuracy:
0.5361 - val_loss: 1.2887 - val_accuracy: 0.6509
Epoch 2/20
29/29 [==============================] - 1s 33ms/step - loss: 0.7250 - accuracy:
0.6868 - val_loss: 2.9556 - val_accuracy: 0.4852
Epoch 3/20
29/29 [==============================] - 1s 33ms/step - loss: 0.6263 - accuracy:
0.7421 - val_loss: 5.7686 - val_accuracy: 0.4398
Epoch 4/20
29/29 [==============================] - 1s 33ms/step - loss: 0.5022 - accuracy:
0.7920 - val_loss: 4.6141 - val_accuracy: 0.4201
Epoch 5/20
29/29 [==============================] - 1s 34ms/step - loss: 0.5188 - accuracy:
0.7890 - val_loss: 1.5335 - val_accuracy: 0.5266
Epoch 6/20
29/29 [==============================] - 1s 33ms/step - loss: 0.4252 - accuracy:
```

```
0.8276 - val_loss: 1.8186 - val_accuracy: 0.4497
Epoch 7/20
29/29 [==============================] - 1s 31ms/step - loss: 0.4033 - accuracy:
0.8360 - val_loss: 0.7950 - val_accuracy: 0.6548
Epoch 8/20
29/29 [==============================] - 1s 33ms/step - loss: 0.3543 - accuracy:
0.8631 - val_loss: 1.0247 - val_accuracy: 0.5680
Epoch 9/20
29/29 [==============================] - 1s 31ms/step - loss: 0.3218 - accuracy:
0.8730 - val_loss: 1.1533 - val_accuracy: 0.6331
Epoch 10/20
29/29 [==============================] - 1s 33ms/step - loss: 0.3066 - accuracy:
0.8715 - val_loss: 0.5013 - val_accuracy: 0.7771
Epoch 11/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2405 - accuracy:
0.9130 - val_loss: 0.5356 - val_accuracy: 0.7613
Epoch 12/20
29/29 [==============================] - 1s 31ms/step - loss: 0.2388 - accuracy:
0.9096 - val_loss: 0.5823 - val_accuracy: 0.7574
Epoch 13/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2262 - accuracy:
0.9130 - val_loss: 0.3605 - val_accuracy: 0.8481
Epoch 14/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1719 - accuracy:
0.9457 - val_loss: 0.5022 - val_accuracy: 0.7968
Epoch 15/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1490 - accuracy:
0.9486 - val_loss: 0.3855 - val_accuracy: 0.8442
Epoch 16/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1676 - accuracy:
0.9461 - val_loss: 0.3485 - val_accuracy: 0.8462
Epoch 17/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1519 - accuracy:
0.9481 - val_loss: 0.3748 - val_accuracy: 0.8501
Epoch 18/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1397 - accuracy:
0.9550 - val_loss: 0.3741 - val_accuracy: 0.8560
Epoch 19/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1266 - accuracy:
0.9575 - val_loss: 0.3493 - val_accuracy: 0.8600
Epoch 20/20
29/29 [==============================] - 1s 31ms/step - loss: 0.0917 - accuracy:
0.9773 - val_loss: 0.3574 - val_accuracy: 0.8540
{'loss': [1.5736877918243408, 0.7249838709831238, 0.6262670755386353,
0.5021548867225647, 0.5188460946083069, 0.4252202808856964, 0.40334567427635193,
0.35426923632621765, 0.3218279480934143, 0.30658549070358276,
0.2404756397008896, 0.23880411684513092, 0.22619786858558655,
0.17187747359275818, 0.14896313846111298, 0.16756878793239594,
```

0.15194889903068542, 0.13968631625175476, 0.12658414244651794,
0.0916784480214119], 'accuracy': [0.5360671877861023, 0.6867588758468628,
0.742094874382019, 0.7919960618019104, 0.7890316247940063, 0.8275691866874695,
0.8359683752059937, 0.8631423115730286, 0.8730236887931824, 0.8715415000915527,
0.9130434989929199, 0.9095849990844727, 0.9130434989929199, 0.945652186870575,
0.948616623878479, 0.9461462497711182, 0.948122501373291, 0.9550395011901855,
0.9575098752975464, 0.9772727489471436], 'val_loss': [1.2886912822723389,
2.9555747509002686, 5.768579483032227, 4.61414098739624, 1.5334751605987549,
1.8185975551605225, 0.795012891292572, 1.0246777534484863, 1.1532648801803589,
0.5012576580047607, 0.5355802178382874, 0.5822715163230896, 0.3605214059352875,
0.5022302269935608, 0.3855176270008087, 0.348518967628479, 0.37482571601867676,
0.37408047914505005, 0.34928834438323975, 0.3574080169200897], 'val_accuracy':
[0.6508875489234924, 0.48520711064338684, 0.43984219431877136,
0.4201183319091797, 0.526627242565155, 0.4497041404247284, 0.6548323631286621,
0.5680473446846008, 0.6331360936164856, 0.7771202921867371, 0.7613412141799927,
0.7573964595794678, 0.848126232624054, 0.7968441843986511, 0.844181478023529,
0.8461538553237915, 0.8500986099243164, 0.8560158014297485, 0.8599605560302734,
0.8540433645248413]}

********** Optimizer: <class
'tensorflow.python.keras.optimizer_v2.rmsprop.RMSprop'>
**********
Model: "sequential_8"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| bn0 (BatchNormalization) | (None, 65536) | 262144 |
| activation_16 (Activation) | (None, 65536) | 0 |
| dense_16 (Dense) | (None, 100) | 6553700 |
| activation_17 (Activation) | (None, 100) | 0 |
| dense_17 (Dense) | (None, 3) | 303 |

Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072

Epoch 1/20
29/29 [==============================] - 1s 43ms/step - loss: 67.7466 -
accuracy: 0.5321 - val_loss: 1.0961 - val_accuracy: 0.3077
Epoch 2/20
29/29 [==============================] - 1s 34ms/step - loss: 1.0905 - accuracy:
0.4916 - val_loss: 1.0781 - val_accuracy: 0.3077
Epoch 3/20

```
29/29 [==============================] - 1s 34ms/step - loss: 1.7945 - accuracy:
0.4802 - val_loss: 292.1182 - val_accuracy: 0.4655
Epoch 4/20
29/29 [==============================] - 1s 35ms/step - loss: 3.2176 - accuracy:
0.4629 - val_loss: 1.0855 - val_accuracy: 0.4694
Epoch 5/20
29/29 [==============================] - 1s 34ms/step - loss: 1.1662 - accuracy:
0.4975 - val_loss: 1.1058 - val_accuracy: 0.4714
Epoch 6/20
29/29 [==============================] - 1s 33ms/step - loss: 1.1552 - accuracy:
0.4827 - val_loss: 1.1245 - val_accuracy: 0.3057
Epoch 7/20
29/29 [==============================] - 1s 34ms/step - loss: 1.2096 - accuracy:
0.5257 - val_loss: 1.1365 - val_accuracy: 0.4813
Epoch 8/20
29/29 [==============================] - 1s 34ms/step - loss: 0.8641 - accuracy:
0.5894 - val_loss: 1.0544 - val_accuracy: 0.5838
Epoch 9/20
29/29 [==============================] - 1s 34ms/step - loss: 0.8495 - accuracy:
0.5509 - val_loss: 1.1414 - val_accuracy: 0.3235
Epoch 10/20
29/29 [==============================] - 1s 34ms/step - loss: 0.9076 - accuracy:
0.6028 - val_loss: 1.0787 - val_accuracy: 0.5266
Epoch 11/20
29/29 [==============================] - 1s 35ms/step - loss: 1.6699 - accuracy:
0.6023 - val_loss: 1.0812 - val_accuracy: 0.4615
Epoch 12/20
29/29 [==============================] - 1s 36ms/step - loss: 0.7086 - accuracy:
0.7026 - val_loss: 1.1067 - val_accuracy: 0.4852
Epoch 13/20
29/29 [==============================] - 1s 34ms/step - loss: 0.7012 - accuracy:
0.7055 - val_loss: 0.9132 - val_accuracy: 0.6410
Epoch 14/20
29/29 [==============================] - 1s 34ms/step - loss: 0.6999 - accuracy:
0.6986 - val_loss: 0.9253 - val_accuracy: 0.5819
Epoch 15/20
29/29 [==============================] - 1s 34ms/step - loss: 0.6503 - accuracy:
0.7327 - val_loss: 1.0782 - val_accuracy: 0.6943
Epoch 16/20
29/29 [==============================] - 1s 34ms/step - loss: 0.7243 - accuracy:
0.7352 - val_loss: 1.3663 - val_accuracy: 0.5503
Epoch 17/20
29/29 [==============================] - 1s 34ms/step - loss: 0.5114 - accuracy:
0.7895 - val_loss: 1.4382 - val_accuracy: 0.7318
Epoch 18/20
29/29 [==============================] - 1s 34ms/step - loss: 0.6315 - accuracy:
0.7708 - val_loss: 1.0125 - val_accuracy: 0.7613
Epoch 19/20
```

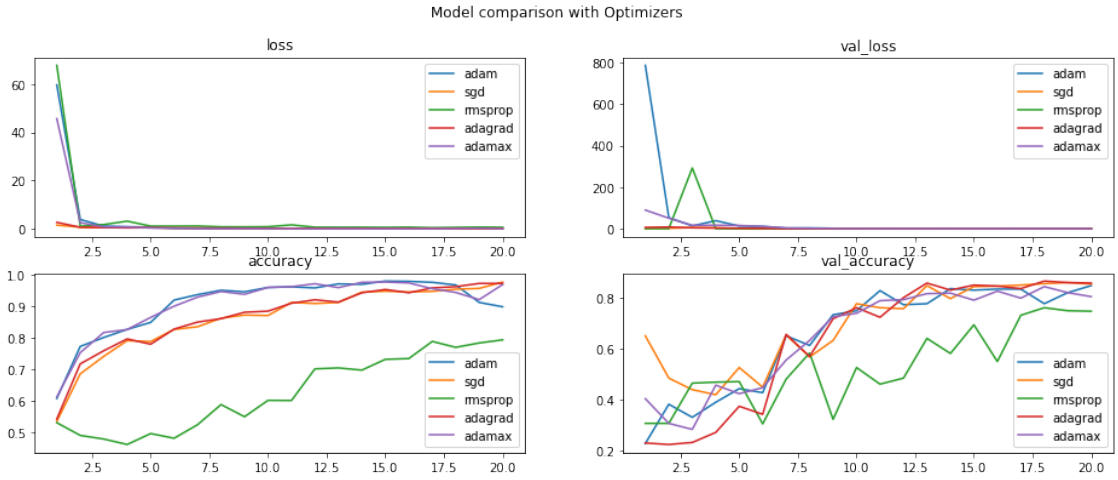
```
29/29 [==============================] - 1s 35ms/step - loss: 0.7513 - accuracy:
0.7846 - val_loss: 1.0293 - val_accuracy: 0.7495
Epoch 20/20
29/29 [==============================] - 1s 33ms/step - loss: 0.6696 - accuracy:
0.7945 - val_loss: 1.1113 - val_accuracy: 0.7475
{'loss': [67.7466049194336, 1.090451717376709, 1.7945061922073364,
3.2176098823547363, 1.1661757230758667, 1.1551953554153442, 1.2095706462860107,
0.8641129732131958, 0.8495026230812073, 0.9075559377670288, 1.669857382774353,
0.7085891962051392, 0.7012041807174683, 0.6999037265777588, 0.6503405570983887,
0.724301815032959, 0.5114284157752991, 0.6314806342124939, 0.7512964010238647,
0.6695606112480164], 'accuracy': [0.5321146249771118, 0.49160078167915344,
0.4802371561527252, 0.4629446566104889, 0.49752965569496155,
0.48270750045776367, 0.5256916880607605, 0.5894268751144409, 0.5508893132209778,
0.6027668118476868, 0.6022727489471436, 0.7025691866874695, 0.7055336236953735,
0.698616623878479, 0.7327075004577637, 0.7351778745651245, 0.7895256876945496,
0.7707509994506836, 0.7845849990844727, 0.7944663763046265], 'val_loss':
[1.0961055755615234, 1.0780812501907349, 292.1181945800781, 1.0855333805084229,
1.105831503868103, 1.1244521141052246, 1.1365278959274292, 1.0543746948242188,
1.141357183456421, 1.0786992311477661, 1.0811880826950073, 1.106706142425537,
0.9132261276245117, 0.9253085851669312, 1.0781821012496948, 1.3662962913513184,
1.4381831884384155, 1.0124512910842896, 1.0293350219726562, 1.1113337278366089],
'val_accuracy': [0.3076923191547394, 0.3076923191547394, 0.46548324823379517,
0.46942800283432007, 0.4714003801345825, 0.30571991205215454,
0.48126232624053955, 0.5838264226913452, 0.32347139716148376, 0.526627242565155,
0.4615384638309479, 0.48520711064338684, 0.6410256624221802, 0.5818540453910828,
0.6942800879478455, 0.5502958297729492, 0.7317554354667664, 0.7613412141799927,
0.7495068907737732, 0.7475345134735107]}


 ************************************************** Optimizer: <class
'tensorflow.python.keras.optimizer_v2.adagrad.Adagrad'>
**************************************************
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
activation_18 (Activation)   (None, 65536)             0
_________________________________________________________________
dense_18 (Dense)             (None, 100)               6553700
_________________________________________________________________
activation_19 (Activation)   (None, 100)               0
_________________________________________________________________
dense_19 (Dense)             (None, 3)                 303
=================================================================
Total params: 6,816,147
Trainable params: 6,685,075
```

```
Non-trainable params: 131,072
_________________________________________________________________
Epoch 1/20
29/29 [==============================] - 1s 42ms/step - loss: 2.7196 - accuracy:
0.5435 - val_loss: 7.0242 - val_accuracy: 0.2308
Epoch 2/20
29/29 [==============================] - 1s 32ms/step - loss: 0.6701 - accuracy:
0.7184 - val_loss: 8.8322 - val_accuracy: 0.2249
Epoch 3/20
29/29 [==============================] - 1s 33ms/step - loss: 0.5548 - accuracy:
0.7599 - val_loss: 5.0920 - val_accuracy: 0.2327
Epoch 4/20
29/29 [==============================] - 1s 33ms/step - loss: 0.4693 - accuracy:
0.7974 - val_loss: 4.1624 - val_accuracy: 0.2722
Epoch 5/20
29/29 [==============================] - 1s 34ms/step - loss: 0.5007 - accuracy:
0.7806 - val_loss: 3.1702 - val_accuracy: 0.3748
Epoch 6/20
29/29 [==============================] - 1s 32ms/step - loss: 0.4027 - accuracy:
0.8286 - val_loss: 2.0239 - val_accuracy: 0.3432
Epoch 7/20
29/29 [==============================] - 1s 33ms/step - loss: 0.3551 - accuracy:
0.8508 - val_loss: 0.8229 - val_accuracy: 0.6568
Epoch 8/20
29/29 [==============================] - 1s 32ms/step - loss: 0.3385 - accuracy:
0.8617 - val_loss: 1.0866 - val_accuracy: 0.5700
Epoch 9/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2939 - accuracy:
0.8819 - val_loss: 0.6260 - val_accuracy: 0.7179
Epoch 10/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2796 - accuracy:
0.8854 - val_loss: 0.5258 - val_accuracy: 0.7613
Epoch 11/20
29/29 [==============================] - 1s 32ms/step - loss: 0.2310 - accuracy:
0.9106 - val_loss: 0.6168 - val_accuracy: 0.7239
Epoch 12/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2139 - accuracy:
0.9214 - val_loss: 0.4591 - val_accuracy: 0.8008
Epoch 13/20
29/29 [==============================] - 1s 32ms/step - loss: 0.2179 - accuracy:
0.9140 - val_loss: 0.3514 - val_accuracy: 0.8580
Epoch 14/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1660 - accuracy:
0.9437 - val_loss: 0.4274 - val_accuracy: 0.8304
Epoch 15/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1485 - accuracy:
0.9541 - val_loss: 0.3885 - val_accuracy: 0.8501
Epoch 16/20
```

```
29/29 [==============================] - 1s 33ms/step - loss: 0.1617 - accuracy:
0.9437 - val_loss: 0.3796 - val_accuracy: 0.8462
Epoch 17/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1358 - accuracy:
0.9590 - val_loss: 0.4111 - val_accuracy: 0.8363
Epoch 18/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1261 - accuracy:
0.9629 - val_loss: 0.3507 - val_accuracy: 0.8659
Epoch 19/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1077 - accuracy:
0.9733 - val_loss: 0.3626 - val_accuracy: 0.8600
Epoch 20/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1025 - accuracy:
0.9733 - val_loss: 0.3625 - val_accuracy: 0.8580
{'loss': [2.719622850418091, 0.6701390743255615, 0.5548432469367981,
0.46925628185272217, 0.5006811618804932, 0.4027077853679657,
0.35511332750320435, 0.33850398659706116, 0.29386869072914124,
0.2796453535556793, 0.23102299869060516, 0.21387340128421783,
0.21791137754917145, 0.1660407930612564, 0.14850583672523499,
0.16167235374450684, 0.13584676384925842, 0.1260766237974167,
0.10773951560258865, 0.10246222466230392], 'accuracy': [0.54347825050354,
0.7183794379234314, 0.7598814368247986, 0.7974308133125305, 0.7806324362754822,
0.8285573124885559, 0.8507905006408691, 0.8616600632667542, 0.8819169998168945,
0.8853754997253418, 0.9105731248855591, 0.9214426875114441, 0.9140316247940063,
0.9436758756637573, 0.9540513753890991, 0.9436758756637573, 0.9589921236038208,
0.9629446864128113, 0.9733201861381531, 0.9733201861381531], 'val_loss':
[7.024216651916504, 8.832229614257812, 5.091989994049072, 4.162367820739746,
3.1701927185058594, 2.0238661766052246, 0.8229429721832275, 1.0866237878799438,
0.6259713172912598, 0.5257624387741089, 0.6168336272239685, 0.45911532640457153,
0.3514207601547241, 0.42740944027900696, 0.38848885893821716,
0.37958192825317383, 0.41107237339019775, 0.35067224502563477,
0.3626026511192322, 0.3624647557735443], 'val_accuracy': [0.23076923191547394,
0.2248520702123642, 0.23274162411689758, 0.27218934893608093,
0.3747534453868866, 0.34319525957107544, 0.6568047404289246, 0.5700197219848633,
0.7179487347602844, 0.7613412141799927, 0.7238658666610718, 0.800788938999176,
0.857988178730011, 0.8303747773170471, 0.8500986099243164, 0.8461538553237915,
0.8362919092178345, 0.8658776879310608, 0.8599605560302734, 0.857988178730011]}


 ************************************************** Optimizer: <class
'tensorflow.python.keras.optimizer_v2.adamax.Adamax'>
**************************************************
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
```

```
activation_20 (Activation)   (None, 65536)            0
_________________________________________________________________
dense_20 (Dense)             (None, 100)               6553700
_________________________________________________________________
activation_21 (Activation)   (None, 100)               0
_________________________________________________________________
dense_21 (Dense)             (None, 3)                 303
=================================================================
Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072
_________________________________________________________________
Epoch 1/20
29/29 [==============================] - 4s 136ms/step - loss: 45.6838 -
accuracy: 0.6136 - val_loss: 90.2035 - val_accuracy: 0.4043
Epoch 2/20
29/29 [==============================] - 1s 33ms/step - loss: 2.5837 - accuracy:
0.7540 - val_loss: 51.1250 - val_accuracy: 0.3077
Epoch 3/20
29/29 [==============================] - 1s 32ms/step - loss: 0.8582 - accuracy:
0.8177 - val_loss: 17.1979 - val_accuracy: 0.2840
Epoch 4/20
29/29 [==============================] - 1s 33ms/step - loss: 0.7894 - accuracy:
0.8271 - val_loss: 17.5431 - val_accuracy: 0.4576
Epoch 5/20
29/29 [==============================] - 1s 33ms/step - loss: 0.4192 - accuracy:
0.8656 - val_loss: 16.1784 - val_accuracy: 0.4241
Epoch 6/20
29/29 [==============================] - 1s 33ms/step - loss: 0.2794 - accuracy:
0.9012 - val_loss: 11.3802 - val_accuracy: 0.4477
Epoch 7/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1959 - accuracy:
0.9298 - val_loss: 4.3736 - val_accuracy: 0.5562
Epoch 8/20
29/29 [==============================] - 1s 34ms/step - loss: 0.1327 - accuracy:
0.9476 - val_loss: 2.7783 - val_accuracy: 0.6331
Epoch 9/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1509 - accuracy:
0.9387 - val_loss: 1.6764 - val_accuracy: 0.7258
Epoch 10/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1217 - accuracy:
0.9605 - val_loss: 1.2759 - val_accuracy: 0.7396
Epoch 11/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1020 - accuracy:
0.9629 - val_loss: 0.9490 - val_accuracy: 0.7890
Epoch 12/20
29/29 [==============================] - 1s 35ms/step - loss: 0.0784 - accuracy:
0.9723 - val_loss: 0.9123 - val_accuracy: 0.7929
```

```
Epoch 13/20
29/29 [==============================] - 1s 48ms/step - loss: 0.1116 - accuracy:
0.9600 - val_loss: 0.8409 - val_accuracy: 0.8166
Epoch 14/20
29/29 [==============================] - 1s 32ms/step - loss: 0.0675 - accuracy:
0.9768 - val_loss: 0.8355 - val_accuracy: 0.8185
Epoch 15/20
29/29 [==============================] - 1s 33ms/step - loss: 0.0682 - accuracy:
0.9778 - val_loss: 1.0515 - val_accuracy: 0.7909
Epoch 16/20
29/29 [==============================] - 1s 32ms/step - loss: 0.0706 - accuracy:
0.9748 - val_loss: 0.8508 - val_accuracy: 0.8264
Epoch 17/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1153 - accuracy:
0.9565 - val_loss: 0.9228 - val_accuracy: 0.7988
Epoch 18/20
29/29 [==============================] - 2s 54ms/step - loss: 0.1314 - accuracy:
0.9447 - val_loss: 0.8500 - val_accuracy: 0.8442
Epoch 19/20
29/29 [==============================] - 1s 34ms/step - loss: 0.2511 - accuracy:
0.9219 - val_loss: 0.8980 - val_accuracy: 0.8205
Epoch 20/20
29/29 [==============================] - 1s 31ms/step - loss: 0.0760 - accuracy:
0.9699 - val_loss: 0.8046 - val_accuracy: 0.8047
{'loss': [45.683834075927734, 2.5836873054504395, 0.8582390546798706,
0.7893685102462769, 0.41918039321899414, 0.2793857455253601, 0.1959441751241684,
0.13267050683498383, 0.15085916221141815, 0.12168104201555252,
0.10203902423381805, 0.0783756673336029, 0.11161989718675613,
0.06745567172765732, 0.06822817772626877, 0.07060226798057556,
0.11533782631158829, 0.13138921558856964, 0.2510664463043213,
0.07596234977245331], 'accuracy': [0.6136363744735718, 0.7539525628089905,
0.8176877498626709, 0.8270751237869263, 0.8656126260757446, 0.9011857509613037,
0.9298418760299683, 0.9476284384727478, 0.9387351870536804, 0.9604743123054504,
0.9629446864128113, 0.9723320007324219, 0.9599802494049072, 0.9767786264419556,
0.9777668118476868, 0.9748023748397827, 0.95652174949646, 0.9446640610694885,
0.9219367504119873, 0.9698616862297058], 'val_loss': [90.20346069335938,
51.124977111816406, 17.19792938232422, 17.54308319091797, 16.17837142944336,
11.380205154418945, 4.373564720153809, 2.7782890796661377, 1.676409125328064,
1.2759265899658203, 0.9489880204200745, 0.9123467206954956, 0.8408997654914856,
0.8354947566986084, 1.0515295267105103, 0.850813627243042, 0.9228474497795105,
0.8500348925590515, 0.898000180721283, 0.804600715637207], 'val_accuracy':
[0.4043392539024353, 0.3076923191547394, 0.2840236723423004, 0.4575936794281006,
0.424063116312027, 0.44773176312446594, 0.5562130212783813, 0.6331360936164856,
0.7258382439613342, 0.7396449446678162, 0.7889546155929565, 0.7928994297981262,
0.8165680766105652, 0.8185404539108276, 0.790926992893219, 0.8264299631118774,
0.7988165616989136, 0.844181478023529, 0.8205128312110901, 0.8047337532043457]}
```




**From above plots I concluded that model with adagrad optimizer, with bias regularization is performing consistent.**

**Learning Rate comparision**

[26]:
```
br = {
    "bias_regularizer": regularizers.l2(1e-4),
}
histories_opt = []
for lr in [
    0.01,
    0.1,
    0.001,
]:
    print("\n\n", "*" * 50, "Learning Rate:", lr, "*" * 50)
    model = create_keras_model(keras.optimizers.Adagrad, br, lr)
    model.summary()
    history = model.fit(
        x=train_xc,
        y=train_yc,
        batch_size=70,
        epochs=20,
        validation_split=0.2,
        verbose=1,
    )
    print(history.history)
    histories_opt.append(history)
plot_comparison(
    histories_opt,
    ["0.01", "0.1", "0.001"],
    "Model comparison with different Learning Rates",
```

```
)
```

```
 ********************************************** Learning Rate: 0.01
**********************************************
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
activation_22 (Activation)   (None, 65536)             0
_________________________________________________________________
dense_22 (Dense)             (None, 100)               6553700
_________________________________________________________________
activation_23 (Activation)   (None, 100)               0
_________________________________________________________________
dense_23 (Dense)             (None, 3)                 303
=================================================================
Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072
_________________________________________________________________
Epoch 1/20
29/29 [==============================] - 1s 41ms/step - loss: 1.6362 - accuracy:
0.5608 - val_loss: 2.7806 - val_accuracy: 0.3452
Epoch 2/20
29/29 [==============================] - 1s 32ms/step - loss: 0.6797 - accuracy:
0.7134 - val_loss: 8.1136 - val_accuracy: 0.2288
Epoch 3/20
29/29 [==============================] - 1s 32ms/step - loss: 0.5444 - accuracy:
0.7722 - val_loss: 7.7082 - val_accuracy: 0.2367
Epoch 4/20
29/29 [==============================] - 1s 33ms/step - loss: 0.4343 - accuracy:
0.8236 - val_loss: 4.6319 - val_accuracy: 0.2781
Epoch 5/20
29/29 [==============================] - 1s 36ms/step - loss: 0.4546 - accuracy:
0.8241 - val_loss: 2.1408 - val_accuracy: 0.4221
Epoch 6/20
29/29 [==============================] - 1s 35ms/step - loss: 0.3591 - accuracy:
0.8607 - val_loss: 2.4416 - val_accuracy: 0.3728
Epoch 7/20
29/29 [==============================] - 1s 34ms/step - loss: 0.3042 - accuracy:
0.8834 - val_loss: 0.6131 - val_accuracy: 0.7337
Epoch 8/20
29/29 [==============================] - 1s 35ms/step - loss: 0.2606 - accuracy:
```

```
0.9037 - val_loss: 1.1356 - val_accuracy: 0.5503
Epoch 9/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1970 - accuracy:
0.9289 - val_loss: 0.6098 - val_accuracy: 0.7574
Epoch 10/20
29/29 [==============================] - 1s 34ms/step - loss: 0.2107 - accuracy:
0.9254 - val_loss: 0.4576 - val_accuracy: 0.8087
Epoch 11/20
29/29 [==============================] - 1s 34ms/step - loss: 0.1426 - accuracy:
0.9565 - val_loss: 0.3740 - val_accuracy: 0.8442
Epoch 12/20
29/29 [==============================] - 1s 35ms/step - loss: 0.1177 - accuracy:
0.9699 - val_loss: 0.3806 - val_accuracy: 0.8304
Epoch 13/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1343 - accuracy:
0.9580 - val_loss: 0.3250 - val_accuracy: 0.8600
Epoch 14/20
29/29 [==============================] - 1s 34ms/step - loss: 0.0941 - accuracy:
0.9758 - val_loss: 0.3459 - val_accuracy: 0.8540
Epoch 15/20
29/29 [==============================] - 1s 33ms/step - loss: 0.0781 - accuracy:
0.9832 - val_loss: 0.3637 - val_accuracy: 0.8619
Epoch 16/20
29/29 [==============================] - 1s 34ms/step - loss: 0.0852 - accuracy:
0.9788 - val_loss: 0.3060 - val_accuracy: 0.8777
Epoch 17/20
29/29 [==============================] - 1s 34ms/step - loss: 0.0695 - accuracy:
0.9852 - val_loss: 0.3428 - val_accuracy: 0.8580
Epoch 18/20
29/29 [==============================] - 1s 34ms/step - loss: 0.0514 - accuracy:
0.9901 - val_loss: 0.3109 - val_accuracy: 0.8876
Epoch 19/20
29/29 [==============================] - 1s 34ms/step - loss: 0.0476 - accuracy:
0.9926 - val_loss: 0.3916 - val_accuracy: 0.8679
Epoch 20/20
29/29 [==============================] - 1s 35ms/step - loss: 0.0405 - accuracy:
0.9960 - val_loss: 0.3210 - val_accuracy: 0.8797
{'loss': [1.6361638307571411, 0.6797153353691101, 0.5444017052650452,
0.4343019723892212, 0.45461997389793396, 0.3591187596321106, 0.3041655123233795,
0.26060691475868225, 0.19696339964866638, 0.21068331599235535,
0.14259649813175201, 0.11766022443771362, 0.13427825272083282,
0.09409086406230927, 0.07810118049383163, 0.0852382630109787,
0.0695108026266098, 0.05140826106071472, 0.04758131876587868,
0.040527258068323135], 'accuracy': [0.5607707500457764, 0.7134387493133545,
0.7722331881523132, 0.823616623878479, 0.8241106867790222, 0.8606719374656677,
0.8833991885185242, 0.9036561250686646, 0.9288537502288818, 0.9253952503204346,
0.95652174949646, 0.9698616862297058, 0.9580039381980896, 0.9757905006408691,
0.9832015633583069, 0.9787549376487732, 0.9851778745651245, 0.9901185631752014,
```

```
0.9925889372825623, 0.9960474371910095], 'val_loss': [2.7805769443511963,
8.113579750061035, 7.7082200050354, 4.63187313079834, 2.1408488750457764,
2.4416396617889404, 0.6130736470222473, 1.1355825662612915, 0.6097692847251892,
0.45757195353507996, 0.3740278482437134, 0.38059353828430176,
0.32495665550231934, 0.3459233343601227, 0.3636811077594757, 0.3060438334941864,
0.3428073227405548, 0.3108973503112793, 0.3916442394256592,
0.32101497054100037], 'val_accuracy': [0.3451676666736603, 0.2287968397140503,
0.23668639361858368, 0.2781065106391907, 0.4220907390117645,
0.37278106808662415, 0.7337278127670288, 0.5502958297729492, 0.7573964595794678,
0.8086785078048706, 0.844181478023529, 0.8303747773170471, 0.8599605560302734,
0.8540433645248413, 0.8619329333305359, 0.8777120113372803, 0.857988178730011,
0.8875739574432373, 0.867850124835968, 0.8796843886375427]}


 ************************************************** Learning Rate: 0.1
**************************************************
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
activation_24 (Activation)   (None, 65536)             0
_________________________________________________________________
dense_24 (Dense)             (None, 100)               6553700
_________________________________________________________________
activation_25 (Activation)   (None, 100)               0
_________________________________________________________________
dense_25 (Dense)             (None, 3)                 303
=================================================================
Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072
_________________________________________________________________
Epoch 1/20
29/29 [==============================] - 1s 41ms/step - loss: 21.2436 -
accuracy: 0.4447 - val_loss: 1.0648 - val_accuracy: 0.4655
Epoch 2/20
29/29 [==============================] - 1s 32ms/step - loss: 1.1859 - accuracy:
0.4545 - val_loss: 1.0567 - val_accuracy: 0.4655
Epoch 3/20
29/29 [==============================] - 1s 32ms/step - loss: 1.0692 - accuracy:
0.4595 - val_loss: 1.0554 - val_accuracy: 0.4655
Epoch 4/20
29/29 [==============================] - 1s 32ms/step - loss: 1.0593 - accuracy:
0.4595 - val_loss: 1.0545 - val_accuracy: 0.4675
Epoch 5/20
29/29 [==============================] - 1s 32ms/step - loss: 1.0558 - accuracy:
```

```
0.4654 - val_loss: 1.0672 - val_accuracy: 0.4734
Epoch 6/20
29/29 [==============================] - 1s 32ms/step - loss: 1.0597 - accuracy:
0.4901 - val_loss: 1.0541 - val_accuracy: 0.4675
Epoch 7/20
29/29 [==============================] - 1s 32ms/step - loss: 1.0587 - accuracy:
0.4600 - val_loss: 1.0531 - val_accuracy: 0.4675
Epoch 8/20
29/29 [==============================] - 1s 32ms/step - loss: 1.0573 - accuracy:
0.4610 - val_loss: 1.0531 - val_accuracy: 0.4675
Epoch 9/20
29/29 [==============================] - 1s 32ms/step - loss: 1.0593 - accuracy:
0.4605 - val_loss: 1.0533 - val_accuracy: 0.4675
Epoch 10/20
29/29 [==============================] - 1s 32ms/step - loss: 1.0614 - accuracy:
0.4580 - val_loss: 1.0531 - val_accuracy: 0.4675
Epoch 11/20
29/29 [==============================] - 1s 33ms/step - loss: 1.0570 - accuracy:
0.4610 - val_loss: 1.0532 - val_accuracy: 0.4675
Epoch 12/20
29/29 [==============================] - 1s 33ms/step - loss: 1.0568 - accuracy:
0.4605 - val_loss: 1.0545 - val_accuracy: 0.4675
Epoch 13/20
29/29 [==============================] - 1s 33ms/step - loss: 1.0570 - accuracy:
0.4605 - val_loss: 1.0551 - val_accuracy: 0.4714
Epoch 14/20
29/29 [==============================] - 1s 33ms/step - loss: 1.0560 - accuracy:
0.4620 - val_loss: 1.0562 - val_accuracy: 0.4714
Epoch 15/20
29/29 [==============================] - 1s 33ms/step - loss: 1.0553 - accuracy:
0.4620 - val_loss: 1.0556 - val_accuracy: 0.4734
Epoch 16/20
29/29 [==============================] - 1s 32ms/step - loss: 1.1038 - accuracy:
0.4625 - val_loss: 1.0575 - val_accuracy: 0.4675
Epoch 17/20
29/29 [==============================] - 1s 33ms/step - loss: 1.0557 - accuracy:
0.4610 - val_loss: 1.0586 - val_accuracy: 0.4675
Epoch 18/20
29/29 [==============================] - 1s 33ms/step - loss: 1.0557 - accuracy:
0.4625 - val_loss: 1.0606 - val_accuracy: 0.4694
Epoch 19/20
29/29 [==============================] - 1s 33ms/step - loss: 1.0556 - accuracy:
0.4620 - val_loss: 1.0608 - val_accuracy: 0.4675
Epoch 20/20
29/29 [==============================] - 1s 33ms/step - loss: 1.0552 - accuracy:
0.4625 - val_loss: 1.0580 - val_accuracy: 0.4694
{'loss': [21.243555068969727, 1.1858861446380615, 1.069240689277649,
1.0592528581619263, 1.0558189153671265, 1.0597106218338013, 1.0587058067321777,
```

1.0573369264602661, 1.0592708587646484, 1.0614147186279297, 1.0570229291915894,
1.0567877292633057, 1.0569671392440796, 1.0560184717178345, 1.055307149887085,
1.1037620306015015, 1.055690884590149, 1.0556987524032593, 1.055572509765625,
1.0552341938018799], 'accuracy': [0.44466403126716614, 0.4545454680919647,
0.4594861567020416, 0.4594861567020416, 0.46541503071784973, 0.4901185631752014,
0.4599802494049072, 0.46096837520599365, 0.46047431230545044,
0.4580039381980896, 0.46096837520599365, 0.46047431230545044,
0.46047431230545044, 0.46195653080940247, 0.46195653080940247,
0.4624505937099457, 0.46096837520599365, 0.4624505937099457,
0.46195653080940247, 0.4624505937099457], 'val_loss': [1.0647611618041992,
1.0566632747650146, 1.055438756942749, 1.0544646978378296, 1.0671641826629639,
1.0540574789047241, 1.0531448125839233, 1.0531481504440308, 1.0532644987106323,
1.053057312965393, 1.0531775951385498, 1.054544448852539, 1.0550636053085327,
1.0561957359313965, 1.0555561780929565, 1.05752694606781, 1.0585614442825317,
1.0606145858764648, 1.060788869857788, 1.0579910278320312], 'val_accuracy':
[0.46548324823379517, 0.46548324823379517, 0.46548324823379517,
0.4674556255340576, 0.47337278723716736, 0.4674556255340576, 0.4674556255340576,
0.4674556255340576, 0.4674556255340576, 0.4674556255340576, 0.4674556255340576,
0.4674556255340576, 0.4714003801345825, 0.4714003801345825, 0.47337278723716736,
0.4674556255340576, 0.4674556255340576, 0.46942800283432007, 0.4674556255340576,
0.46942800283432007]}

********** Learning Rate: 0.001
**********
Model: "sequential_13"

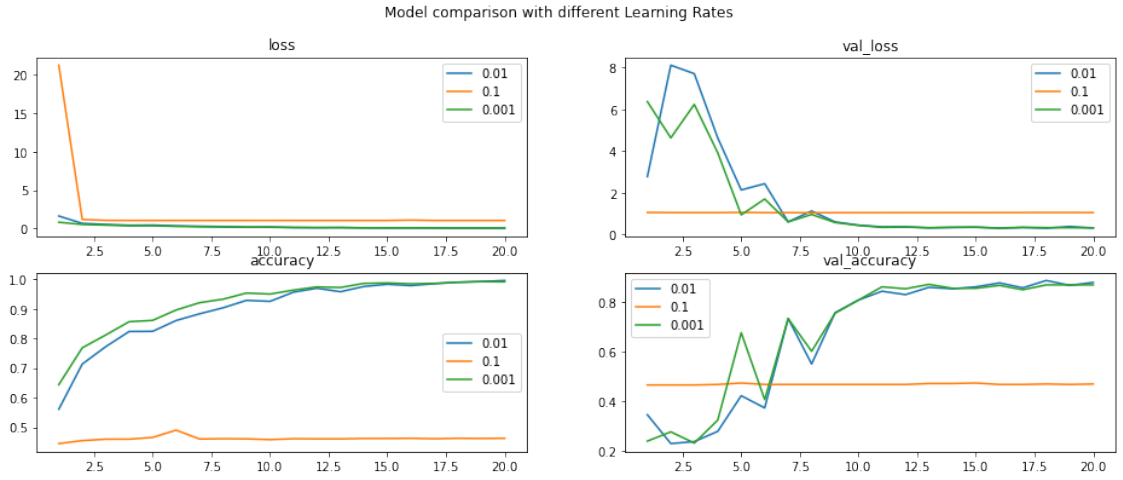
| Layer (type) | Output Shape | Param # |
|---|---|---|
| bn0 (BatchNormalization) | (None, 65536) | 262144 |
| activation_26 (Activation) | (None, 65536) | 0 |
| dense_26 (Dense) | (None, 100) | 6553700 |
| activation_27 (Activation) | (None, 100) | 0 |
| dense_27 (Dense) | (None, 3) | 303 |

Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072

Epoch 1/20
29/29 [==============================] - 1s 41ms/step - loss: 0.8294 - accuracy:
0.6438 - val_loss: 6.3711 - val_accuracy: 0.2387
Epoch 2/20
29/29 [==============================] - 1s 34ms/step - loss: 0.5402 - accuracy:

```
0.7683 - val_loss: 4.6332 - val_accuracy: 0.2761
Epoch 3/20
29/29 [==============================] - 1s 33ms/step - loss: 0.4534 - accuracy:
0.8118 - val_loss: 6.2383 - val_accuracy: 0.2308
Epoch 4/20
29/29 [==============================] - 1s 33ms/step - loss: 0.3656 - accuracy:
0.8567 - val_loss: 3.9197 - val_accuracy: 0.3235
Epoch 5/20
29/29 [==============================] - 1s 32ms/step - loss: 0.3673 - accuracy:
0.8612 - val_loss: 0.9501 - val_accuracy: 0.6765
Epoch 6/20
29/29 [==============================] - 1s 32ms/step - loss: 0.2902 - accuracy:
0.8962 - val_loss: 1.7076 - val_accuracy: 0.4063
Epoch 7/20
29/29 [==============================] - 1s 32ms/step - loss: 0.2422 - accuracy:
0.9209 - val_loss: 0.6148 - val_accuracy: 0.7337
Epoch 8/20
29/29 [==============================] - 1s 32ms/step - loss: 0.2144 - accuracy:
0.9328 - val_loss: 0.9678 - val_accuracy: 0.6016
Epoch 9/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1799 - accuracy:
0.9536 - val_loss: 0.5829 - val_accuracy: 0.7554
Epoch 10/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1800 - accuracy:
0.9501 - val_loss: 0.4423 - val_accuracy: 0.8087
Epoch 11/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1459 - accuracy:
0.9634 - val_loss: 0.3517 - val_accuracy: 0.8619
Epoch 12/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1315 - accuracy:
0.9743 - val_loss: 0.3649 - val_accuracy: 0.8540
Epoch 13/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1342 - accuracy:
0.9723 - val_loss: 0.3333 - val_accuracy: 0.8718
Epoch 14/20
29/29 [==============================] - 1s 33ms/step - loss: 0.1033 - accuracy:
0.9862 - val_loss: 0.3551 - val_accuracy: 0.8560
Epoch 15/20
29/29 [==============================] - 1s 32ms/step - loss: 0.0965 - accuracy:
0.9876 - val_loss: 0.3537 - val_accuracy: 0.8560
Epoch 16/20
29/29 [==============================] - 1s 32ms/step - loss: 0.1003 - accuracy:
0.9847 - val_loss: 0.3181 - val_accuracy: 0.8679
Epoch 17/20
29/29 [==============================] - 1s 32ms/step - loss: 0.0928 - accuracy:
0.9862 - val_loss: 0.3529 - val_accuracy: 0.8501
Epoch 18/20
29/29 [==============================] - 1s 33ms/step - loss: 0.0806 - accuracy:
```

```
0.9901 - val_loss: 0.3368 - val_accuracy: 0.8698
Epoch 19/20
29/29 [==============================] - 1s 32ms/step - loss: 0.0805 - accuracy:
0.9921 - val_loss: 0.3335 - val_accuracy: 0.8698
Epoch 20/20
29/29 [==============================] - 1s 32ms/step - loss: 0.0729 - accuracy:
0.9916 - val_loss: 0.3253 - val_accuracy: 0.8698
{'loss': [0.829406201839447, 0.5402140617370605, 0.453418105840683,
0.36564233899116516, 0.36732518672943115, 0.2901691198348999,
0.2422049343585968, 0.21443039178848267, 0.1799161434173584,
0.17997808754444122, 0.14594250917434692, 0.13148240745067596,
0.13424836099147797, 0.10331176966428757, 0.09651429951190948,
0.10029936581850052, 0.09278703480958939, 0.08059690147638321,
0.08053843677043915, 0.07292195409536362], 'accuracy': [0.643774688243866,
0.7682806253433228, 0.8117588758468628, 0.8567193746566772, 0.8611660003662109,
0.8962450623512268, 0.9209486246109009, 0.9328063130378723, 0.9535573124885559,
0.9500988125801086, 0.9634387493133545, 0.9743083119392395, 0.9723320007324219,
0.9861660003662109, 0.9876482486724854, 0.9846838116645813, 0.9861660003662109,
0.9901185631752014, 0.992094874382019, 0.9916008114814758], 'val_loss':
[6.371115684509277, 4.633245468139648, 6.238282203674316, 3.9197168350219727,
0.9501459002494812, 1.7075889110565186, 0.6147919297218323, 0.9678063988685608,
0.5829147696495056, 0.4422946870326996, 0.3516780138015747, 0.36494868993759155,
0.3332577347755432, 0.35506314039230347, 0.3537406325340271, 0.3181333541870117,
0.35290130972862244, 0.3367893099784851, 0.33354365825653076,
0.3253117501735687], 'val_accuracy': [0.23865877091884613, 0.2761341333389282,
0.23076923191547394, 0.32347139716148376, 0.6765285730361938,
0.40631163120269775, 0.7337278127670288, 0.6015779376029968, 0.7554240822792053,
0.8086785078048706, 0.8619329333305359, 0.8540433645248413, 0.8717948794364929,
0.8560158014297485, 0.8560158014297485, 0.867850124835968, 0.8500986099243164,
0.8698225021362305, 0.8698225021362305, 0.8698225021362305]}
```



### 1.0.30 Train the model by using above parameters

[27]:
```
model = create_keras_model(keras.optimizers.Adagrad, br, learning_rate=0.001)
model.summary()
history = model.fit(
    x=train_xc, y=train_yc, batch_size=70, epochs=50, validation_split=0.2,␣
 ↪verbose=1,
)
```

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
activation_28 (Activation)   (None, 65536)             0
_________________________________________________________________
dense_28 (Dense)             (None, 100)               6553700
_________________________________________________________________
activation_29 (Activation)   (None, 100)               0
_________________________________________________________________
dense_29 (Dense)             (None, 3)                 303
=================================================================
Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072
_________________________________________________________________
Epoch 1/45
29/29 [==============================] - 1s 43ms/step - loss: 1.0411 - accuracy:
0.5879 - val_loss: 9.0706 - val_accuracy: 0.2308
Epoch 2/45
29/29 [==============================] - 1s 32ms/step - loss: 0.6134 - accuracy:
0.7465 - val_loss: 6.6489 - val_accuracy: 0.2288
Epoch 3/45
29/29 [==============================] - 1s 32ms/step - loss: 0.5099 - accuracy:
0.7935 - val_loss: 4.7042 - val_accuracy: 0.2426
Epoch 4/45
29/29 [==============================] - 1s 33ms/step - loss: 0.4328 - accuracy:
0.8295 - val_loss: 3.9449 - val_accuracy: 0.2742
Epoch 5/45
29/29 [==============================] - 1s 32ms/step - loss: 0.4172 - accuracy:
0.8409 - val_loss: 1.7404 - val_accuracy: 0.4536
Epoch 6/45
29/29 [==============================] - 1s 34ms/step - loss: 0.3491 - accuracy:
0.8794 - val_loss: 1.9591 - val_accuracy: 0.4004
Epoch 7/45
29/29 [==============================] - 1s 35ms/step - loss: 0.3114 - accuracy:
0.8928 - val_loss: 0.8174 - val_accuracy: 0.6430
```

```
Epoch 8/45
29/29 [==============================] - 1s 34ms/step - loss: 0.2805 - accuracy:
0.9056 - val_loss: 0.9493 - val_accuracy: 0.5937
Epoch 9/45
29/29 [==============================] - 1s 33ms/step - loss: 0.2468 - accuracy:
0.9219 - val_loss: 0.5923 - val_accuracy: 0.7515
Epoch 10/45
29/29 [==============================] - 1s 33ms/step - loss: 0.2394 - accuracy:
0.9279 - val_loss: 0.5023 - val_accuracy: 0.7771
Epoch 11/45
29/29 [==============================] - 1s 36ms/step - loss: 0.2039 - accuracy:
0.9422 - val_loss: 0.3802 - val_accuracy: 0.8521
Epoch 12/45
29/29 [==============================] - 1s 35ms/step - loss: 0.1853 - accuracy:
0.9545 - val_loss: 0.3915 - val_accuracy: 0.8383
Epoch 13/45
29/29 [==============================] - 1s 34ms/step - loss: 0.1838 - accuracy:
0.9555 - val_loss: 0.3497 - val_accuracy: 0.8619
Epoch 14/45
29/29 [==============================] - 1s 34ms/step - loss: 0.1566 - accuracy:
0.9644 - val_loss: 0.3731 - val_accuracy: 0.8521
Epoch 15/45
29/29 [==============================] - 1s 33ms/step - loss: 0.1457 - accuracy:
0.9748 - val_loss: 0.3704 - val_accuracy: 0.8560
Epoch 16/45
29/29 [==============================] - 1s 35ms/step - loss: 0.1443 - accuracy:
0.9733 - val_loss: 0.3434 - val_accuracy: 0.8659
Epoch 17/45
29/29 [==============================] - 1s 34ms/step - loss: 0.1346 - accuracy:
0.9778 - val_loss: 0.3780 - val_accuracy: 0.8521
Epoch 18/45
29/29 [==============================] - 1s 35ms/step - loss: 0.1207 - accuracy:
0.9832 - val_loss: 0.3346 - val_accuracy: 0.8757
Epoch 19/45
29/29 [==============================] - 1s 33ms/step - loss: 0.1211 - accuracy:
0.9778 - val_loss: 0.3391 - val_accuracy: 0.8698
Epoch 20/45
29/29 [==============================] - 1s 36ms/step - loss: 0.1141 - accuracy:
0.9837 - val_loss: 0.3362 - val_accuracy: 0.8718
Epoch 21/45
29/29 [==============================] - 1s 34ms/step - loss: 0.1131 - accuracy:
0.9812 - val_loss: 0.3400 - val_accuracy: 0.8698
Epoch 22/45
29/29 [==============================] - 1s 35ms/step - loss: 0.0950 - accuracy:
0.9867 - val_loss: 0.3324 - val_accuracy: 0.8757
Epoch 23/45
29/29 [==============================] - 1s 35ms/step - loss: 0.0963 - accuracy:
0.9876 - val_loss: 0.3180 - val_accuracy: 0.8856
```

```
Epoch 24/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0911 - accuracy:
0.9891 - val_loss: 0.3419 - val_accuracy: 0.8659
Epoch 25/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0804 - accuracy:
0.9926 - val_loss: 0.3172 - val_accuracy: 0.8856
Epoch 26/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0728 - accuracy:
0.9941 - val_loss: 0.3196 - val_accuracy: 0.8895
Epoch 27/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0736 - accuracy:
0.9931 - val_loss: 0.3164 - val_accuracy: 0.8856
Epoch 28/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0672 - accuracy:
0.9960 - val_loss: 0.3142 - val_accuracy: 0.8876
Epoch 29/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0655 - accuracy:
0.9951 - val_loss: 0.3297 - val_accuracy: 0.8698
Epoch 30/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0666 - accuracy:
0.9941 - val_loss: 0.3146 - val_accuracy: 0.8817
Epoch 31/45
29/29 [==============================] - 1s 33ms/step - loss: 0.0591 - accuracy:
0.9965 - val_loss: 0.3160 - val_accuracy: 0.8817
Epoch 32/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0646 - accuracy:
0.9960 - val_loss: 0.3089 - val_accuracy: 0.8915
Epoch 33/45
29/29 [==============================] - 1s 35ms/step - loss: 0.0590 - accuracy:
0.9936 - val_loss: 0.3167 - val_accuracy: 0.8876
Epoch 34/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0569 - accuracy:
0.9975 - val_loss: 0.3077 - val_accuracy: 0.8856
Epoch 35/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0541 - accuracy:
0.9965 - val_loss: 0.3077 - val_accuracy: 0.8915
Epoch 36/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0563 - accuracy:
0.9956 - val_loss: 0.3166 - val_accuracy: 0.8698
Epoch 37/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0517 - accuracy:
0.9985 - val_loss: 0.3035 - val_accuracy: 0.8935
Epoch 38/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0455 - accuracy:
0.9960 - val_loss: 0.3016 - val_accuracy: 0.8935
Epoch 39/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0450 - accuracy:
0.9980 - val_loss: 0.3074 - val_accuracy: 0.8935
```

```
Epoch 40/45
29/29 [==============================] - 1s 33ms/step - loss: 0.0480 - accuracy:
0.9960 - val_loss: 0.3107 - val_accuracy: 0.8915
Epoch 41/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0412 - accuracy:
0.9990 - val_loss: 0.3057 - val_accuracy: 0.8856
Epoch 42/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0524 - accuracy:
0.9965 - val_loss: 0.3028 - val_accuracy: 0.8895
Epoch 43/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0453 - accuracy:
0.9980 - val_loss: 0.3137 - val_accuracy: 0.8955
Epoch 44/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0400 - accuracy:
0.9995 - val_loss: 0.3088 - val_accuracy: 0.8974
Epoch 45/45
29/29 [==============================] - 1s 34ms/step - loss: 0.0409 - accuracy:
0.9975 - val_loss: 0.3085 - val_accuracy: 0.8895
```

#### 1.0.31 Final trained model has validation accuracy of approximatley 89% and loss is around 0.30

[28]:
```
result = model.evaluate(test_xc, test_yc, batch_size=70)
```

```
10/10 [==============================] - 0s 11ms/step - loss: 0.3460 - accuracy:
0.8784
```

#### 1.0.32 Test Data also shows approximately 88% Accuracy and loss 0.34

[29]:
```
print(result)
```

```
[0.34597456455230713, 0.8783570528030396]
```

#### 1.0.33 Coompare predicted labels with observed labels

[30]:
```
predictions = model.predict(test_xc)
y_predicted = predictions.argmax(axis=-1)
# print(y_predicted)
y = test_yc[:].argmax(axis=-1)
# print(y)
total_match = 0
for i in range(len(y_predicted)):
    print("Observed=%s, Predicted=%s" % (y_predicted[i], y[i]))
    if y_predicted[i] == y[i]:
        total_match += 1
accuracy = total_match / len(y_predicted)

print("Calculated Accuracy:", accuracy * 100)
```

```
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=0
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=2
Observed=0, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
```

```
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=0
Observed=2, Predicted=1
Observed=2, Predicted=1
Observed=1, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=0
Observed=0, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
```

```
Observed=0, Predicted=0
Observed=1, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=0, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=1
Observed=1, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
```

```
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=0, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
```

```
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=1, Predicted=0
Observed=1, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=0
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
```

```
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=0
Observed=0, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=2
Observed=2, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
```

```
Observed=2, Predicted=2
Observed=2, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
```

```
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=0, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
```

```
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=0
```

```
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=1
Observed=2, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=0, Predicted=1
Observed=0, Predicted=0
Observed=0, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=0
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
```

```
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=2
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=0, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=0, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=0
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
```

```
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=2, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=2, Predicted=2
```

```
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=2
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=2, Predicted=1
Observed=2, Predicted=2
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=1
Observed=1, Predicted=0
Observed=1, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=2, Predicted=2
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=1, Predicted=0
Observed=1, Predicted=1
Observed=2, Predicted=0
Observed=0, Predicted=0
Observed=2, Predicted=2
Observed=2, Predicted=2
```

```
Observed=0, Predicted=0
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=0, Predicted=0
Observed=1, Predicted=1
Observed=1, Predicted=1
Calculated Accuracy: 87.83570300157977
```