Lab Six: CNNs

Member : Yang Shen

Metric

This dataset I used in lab2. [https://www.kaggle.com/vipoooool/new-plant-diseases-dataset (https://www.kaggle.com/vipoooool/new-plant-diseases-dataset)](https://www.kaggle.com/vipoooool/new-plant-diseases-dataset)

This dataset is containing the images of 10 different crops' leaves. For each crop, there are subfiles containing images about healthy and unhealthy leaves. The same leaf will take the photo from different angles. The main purpose of this dataset is to classify different leaves by their categories. We can know the name of the crop and whether the crop is getting the disease by analyzing the leaves photo.

The third party who interested in this result is seeds company. They sell millions of seeds over a year. The quality of seeds is important for the company. Seeds company usually sell some anti-disease seeds in the market. But the performance of anti-disease seeds needs to keep tracking. Farmers can upload the photos to seeds company for daily bases. The seeds company can use our method to find out the category of leaves and what disease the crops got, or crops are healthy. So that they can give advice about how to deal with sick crops to farmers. Also, they can monitor the performance of their seeds.

The performance of our model needs to be accurate in order to find out which disease and plant it is. My metric is accuracy. Becasue my task is prediciton which class the plant is. The accuracy directly reflect what is the ratio of my correction prediction. All other false positive and false negative are meaningless, because if it get wrong, the picture need to check by human. The only correct classification can save time from human checking.

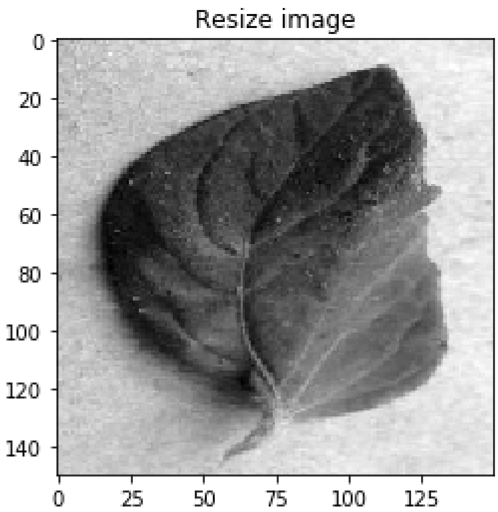
In [1]:

```
#https://www.youtube.com/watch?v=j-3vuBynnOE
%matplotlib inline
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
import os
import cv2
datadir = "C:/Users/jacks/Desktop/plant"
Category = ["Apple_Black_rot","Apple_healthy","Grape_Black_rot","Grape_healthy","Pepper_Bacterial_spot","Pepper_healthy","Potato_healthy","Potato_Late_blight","Tomato_Bacterial_spot","Tomato_healthy"]

for i in Category:
    path = os.path.join(datadir, i) #path to the leaf data
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)# read all the leaves file from each sub file

        break   #show some images not all
```

In [2]:

```
print('original size of the images:',img_array.shape)
print('===================================')
#show the resize of images
imagesize = 150
resize_array = cv2.resize(img_array, (imagesize,imagesize))
plt.imshow(resize_array, cmap = 'gray')
plt.title('Resize image')
plt.show()
```

```
original size of the images: (256, 256)
===================================
```



In [3]:

```
images=[] #just reimport the data from the files and resize the images as well
def creat_data():
    for i in Category:
        path = os.path.join(datadir, i)
        class_num = Category.index(i) #set a index for the column which will help me to set the
 names for different leaves
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)
            resize_array = cv2.resize(img_array, (imagesize,imagesize))
            images.append([resize_array, class_num])# append the 1-d images data and the column
 index

creat_data()
```

In [4]:

```
#create the array contain the images data with 1-D image features
X = []
y = []
#seprate the 1-d images data and column values
for features, label in images:
    X.append(features)
    y.append(label)

names = y.copy()
#reshape the array into the correct format, which each row represent one images
X_images = np.array(X).reshape(-1,imagesize,imagesize)
X = np.array(X_images).reshape(len(images),imagesize*imagesize)

_, h, img_wh = X_images.shape
```

In [5]:

```
#https://stackoverflow.com/questions/2582138/finding-and-replacing-elements-in-a-list-python
# replace the index value to the names of leaf
item_to_replace = 0
replacement_value = "Apple_Black_rot" #name of leaf
indices_to_replace = [i for i,x in enumerate(names) if x==item_to_replace]

for i in indices_to_replace:
    names[i] = replacement_value

item_to_replace = 1
replacement_value = "Apple_healthy"
indices_to_replace = [i for i,x in enumerate(names) if x==item_to_replace]

for i in indices_to_replace:
    names[i] = replacement_value

item_to_replace = 2
replacement_value = "Grape_Black_rot"
indices_to_replace = [i for i,x in enumerate(names) if x==item_to_replace]

for i in indices_to_replace:
    names[i] = replacement_value

item_to_replace = 3
replacement_value = "Grape_healthy"
indices_to_replace = [i for i,x in enumerate(names) if x==item_to_replace]

for i in indices_to_replace:
    names[i] = replacement_value

item_to_replace = 4
replacement_value = "Pepper_Bacterial_spot"
indices_to_replace = [i for i,x in enumerate(names) if x==item_to_replace]

for i in indices_to_replace:
    names[i] = replacement_value

item_to_replace = 5
replacement_value = "Pepper_healthy"
indices_to_replace = [i for i,x in enumerate(names) if x==item_to_replace]

for i in indices_to_replace:
    names[i] = replacement_value

item_to_replace = 6
replacement_value = "Potato_healthy"
indices_to_replace = [i for i,x in enumerate(names) if x==item_to_replace]

for i in indices_to_replace:
    names[i] = replacement_value

item_to_replace = 7
replacement_value = "Potato_Late_blight"
indices_to_replace = [i for i,x in enumerate(names) if x==item_to_replace]

for i in indices_to_replace:
    names[i] = replacement_value

item_to_replace = 8
```

```
replacement_value = "Tomato_Bacterial_spot"
indices_to_replace = [i for i,x in enumerate(names) if x==item_to_replace]

for i in indices_to_replace:
    names[i] = replacement_value

item_to_replace = 9
replacement_value = "Tomato_healthy"
indices_to_replace = [i for i,x in enumerate(names) if x==item_to_replace]

for i in indices_to_replace:
    names[i] = replacement_value

names = np.array(names)
y = np.array(y)
```

In [6]:

```
import keras
from keras.models import Sequential
from keras.layers import Reshape
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping
from keras.regularizers import l2
from keras.layers import average
from keras.models import Input, Model

keras.__version__
```

Using TensorFlow backend.

Out[6]:

'2.2.4'

In [7]:

```
from sklearn import metrics as mt
from matplotlib import pyplot as plt
from skimage.io import imshow
import seaborn as sns
%matplotlib inline

def summarize_net(net, X_test, y_test, title_text=''):
    plt.figure(figsize=(15,5))
    yhat = np.argmax(net.predict(X_test), axis=1)
    acc = mt.accuracy_score(y_test,yhat)
    cm = mt.confusion_matrix(y_test,yhat)
    cm = cm/np.sum(cm,axis=1)[:,np.newaxis]
    sns.heatmap(cm, annot=True, fmt='.2f')
    plt.title(title_text+'{:.4f}'.format(acc))
```

In [8]:

```
import os
import struct
import numpy as np
from sklearn.model_selection import train_test_split

NUM_CLASSES = 10
#normalized between -0.5 and 0.5
#https://stackoverflow.com/questions/38025838/normalizing-images-in-opencv
#normalized X, between 0 and 1
X = cv2.normalize(X, None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
X = np.expand_dims(X.reshape((-1,img_wh,img_wh)), axis=3)
y_ohe = keras.utils.to_categorical(y, NUM_CLASSES)

#8/2 split here just for testing if the model is working or not. I will use kfold later
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# NEW: Let's start by fixing the sizes
X_train = np.expand_dims(X_train.reshape((-1,img_wh,img_wh)), axis=3)
X_test = np.expand_dims(X_test.reshape((-1,img_wh,img_wh)), axis=3)
# the image data has been resized to (samples,image_rows,image_cols,image_channels)

# and one hot encoding the output values
y_train_ohe = keras.utils.to_categorical(y_train, NUM_CLASSES)
y_test_ohe = keras.utils.to_categorical(y_test, NUM_CLASSES)


print('New Shape: Rows: %d, image size: (%d,%d,%d)' % (X_train.shape[0], X_train.shape[1], X_tr
ain.shape[2], X_train.shape[3] ))
```

```
New Shape: Rows: 3384, image size: (150,150,1)
```

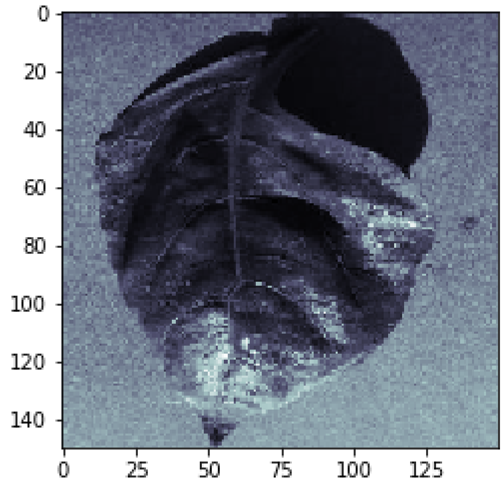
Dividing training and testing

I will use k-fold becasue I only have 4300 rows in my dataset. I think 4300 is not enough to represent the features of dataset. I used 5 fold StratifiedKFold, because I need to use cross validation method and also I want all my plants have the same ratio in each fold. I used both 80/20 and StratifiedKFold as I need. 80/20 split is just for testing the model if it work.

In [9]:

```
print(X_train.shape)
plt.subplot(1,1,1)
plt.imshow(X_train[1].squeeze(),cmap='bone')
plt.show()
```

(3384, 150, 150, 1)



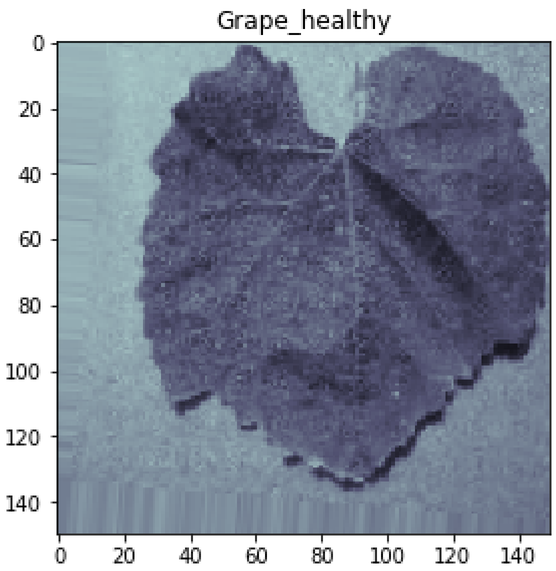
Data expansion

In [10]:

```
#https://keras.io/preprocessing/image/
classes = ['Apple_Black_rot',
           'Apple_healthy',
           'Grape_Black_rot',
           'Grape_healthy',
           'Pepper_Bacterial_spot',
           'Pepper_healthy',
           'Potato_healthy',
           'Potato_Late_blight',
           'Tomato_Bacterial_spot',
           'Tomato_healthy']

datagen = ImageDataGenerator(featurewise_center=False,#Set input mean to 0 over the dataset No
    samplewise_center=False,#Set each sample mean to 0. No
    featurewise_std_normalization=False,#Divide inputs by std of the dataset,i already normaliz
ed my self
    samplewise_std_normalization=False,#i already normalized my self
    zca_whitening=False,#no need to Apply ZCA whitening.
    rotation_range=5, # used, Int. Degree range for random rotations. the dataset is leaves, so
 rotation is useful
    width_shift_range=0.1, # used, Float (fraction of total width). leave shift width is still l
eave,
    height_shift_range=0.1, # used,  Float (fraction of total height). leave shift height is sti
ll leave,
                             #changing width and height is useful, leave is still leave.
    shear_range=0., # Float. Shear Intensity, No shear, prevent to cut important part.
    zoom_range=0.,#No
    channel_shift_range=0.,
    fill_mode='nearest',#Points outside the boundaries of the input are filled, useful here, fil
l the picture
    cval=0.,
    horizontal_flip=True,#horizontal flip a leave is still leave, yes
    vertical_flip=True,#vertical flip a leave is still leave, yes, the leave have unique shape
 to reconized, not like t-shirt
    rescale=None)# no rescale

datagen.fit(X_train)

idx = 0
```

In [120]:

```
tmps = datagen.flow(X_train, y_train_ohe, batch_size=1)

for tmp in tmps:
    imshow(tmp[0].squeeze(),cmap='bone')
    plt.title(classes[np.argmax(tmp[1])])
    break
```



Model1 3X3 kernal size

In [11]:

```
# what if we just want to use the validation data??
from keras.callbacks import EarlyStopping
from keras.regularizers import l2
l2_lambda = 0.0001


def create_cnn1():
    # Use Kaiming He to regularize ReLU layers: https://arxiv.org/pdf/1502.01852.pdf
    # Use Glorot/Bengio for linear/sigmoid/softmax: http://proceedings.mlr.press/v9/glorot10a/gl
orot10a.pdf
    cnn1 = Sequential()

    cnn1.add(Conv2D(filters=32,
                    input_shape = (img_wh,img_wh,1),
                    kernel_size=(3,3),
                    kernel_initializer='he_uniform',
                    kernel_regularizer=l2(l2_lambda),
                    padding='same',
                    activation='relu',
                    data_format="channels_last")) # more compact syntax

    cnn1.add(Conv2D(filters=32,
                kernel_size=(3,3),
                kernel_initializer='he_uniform',
                kernel_regularizer=l2(l2_lambda),
                padding='same',
                activation='relu',data_format="channels_last"))
    cnn1.add(MaxPooling2D(pool_size=(2, 2), data_format="channels_last"))

    cnn1.add(Conv2D(filters=64,
                input_shape = (img_wh,img_wh,1),
                kernel_size=(3,3),
                kernel_initializer='he_uniform',
                kernel_regularizer=l2(l2_lambda),
                padding='same',
                activation='relu',data_format="channels_last")) # more compact syntax

    cnn1.add(Conv2D(filters=64,
                kernel_size=(3,3),
                kernel_initializer='he_uniform',
                kernel_regularizer=l2(l2_lambda),
                padding='same',
                activation='relu'))
    cnn1.add(MaxPooling2D(pool_size=(2, 2), data_format="channels_last"))

    cnn1.add(Conv2D(filters=128,
                input_shape = (img_wh,img_wh,1),
                kernel_size=(3,3),
                kernel_initializer='he_uniform',
                kernel_regularizer=l2(l2_lambda),
                padding='same',
                activation='relu',data_format="channels_last")) # more compact syntax

    cnn1.add(Conv2D(filters=128,
                kernel_size=(3,3),
                kernel_initializer='he_uniform',
                kernel_regularizer=l2(l2_lambda),
                padding='same',
                activation='relu',data_format="channels_last"))
```

```
    # add one layer on flattened output

    cnn1.add(Flatten())
    cnn1.add(Dropout(0.25)) # add some dropout for regularization after conv layers
    cnn1.add(Dense(128,
              activation='relu',
              kernel_initializer='he_uniform',
              kernel_regularizer=l2(l2_lambda)
        ))
    cnn1.add(Dropout(0.5)) # add some dropout for regularization, again!
    cnn1.add(Dense(NUM_CLASSES,
                   activation='softmax',
                   kernel_initializer='glorot_uniform',
                   kernel_regularizer=l2(l2_lambda)
                  ))

    cnn1.compile(loss='categorical_crossentropy', # 'categorical_crossentropy' 'mean_squared_err
or'
                 optimizer='rmsprop', # 'adadelta' 'rmsprop'
                 metrics=['accuracy'])
    return cnn1

cnn1 = create_cnn1()
```

```
WARNING:tensorflow:From D:\APP\conda\lib\site-packages\tensorflow\python\framework
\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is de
precated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\APP\conda\lib\site-packages\keras\backend\tensorflow_ba
ckend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob
is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_p
rob`.
```

In [15]:

```
# the flow method yields batches of images indefinitely, with the given transofmrations
history1 = cnn1.fit_generator(datagen.flow(X_train, y_train_ohe, batch_size=128),
                  steps_per_epoch=int(len(X_train)/128), # how many generators to go through per
epoch
                  epochs=50, verbose=1,
                  validation_data=(X_test,y_test_ohe),
                  callbacks=[EarlyStopping(monitor='val_loss', patience=4)]
                 )
```

```
Epoch 1/50
26/26 [==============================] - 6s 247ms/step - loss: 14.1760 - acc: 0.10
16 - val_loss: 14.4989 - val_acc: 0.1063
Epoch 2/50
26/26 [==============================] - 4s 159ms/step - loss: 11.2625 - acc: 0.10
92 - val_loss: 2.3826 - val_acc: 0.1086
Epoch 3/50
26/26 [==============================] - 4s 158ms/step - loss: 2.4149 - acc: 0.112
6 - val_loss: 2.3620 - val_acc: 0.1063
Epoch 4/50
26/26 [==============================] - 4s 158ms/step - loss: 2.3737 - acc: 0.105
9 - val_loss: 2.3093 - val_acc: 0.1558
Epoch 5/50
26/26 [==============================] - 4s 158ms/step - loss: 2.4042 - acc: 0.143
6 - val_loss: 2.3733 - val_acc: 0.0708
Epoch 6/50
26/26 [==============================] - 4s 159ms/step - loss: 2.2614 - acc: 0.165
1 - val_loss: 2.0086 - val_acc: 0.2149
Epoch 7/50
26/26 [==============================] - 4s 158ms/step - loss: 2.2139 - acc: 0.201
8 - val_loss: 2.1860 - val_acc: 0.2692
Epoch 8/50
26/26 [==============================] - 4s 158ms/step - loss: 2.1762 - acc: 0.212
2 - val_loss: 2.0781 - val_acc: 0.2373
Epoch 9/50
26/26 [==============================] - 4s 158ms/step - loss: 2.0454 - acc: 0.256
3 - val_loss: 1.7894 - val_acc: 0.3577
Epoch 10/50
26/26 [==============================] - 4s 158ms/step - loss: 2.1430 - acc: 0.255
3 - val_loss: 1.6674 - val_acc: 0.4097
Epoch 11/50
26/26 [==============================] - 4s 158ms/step - loss: 1.9088 - acc: 0.307
2 - val_loss: 1.5322 - val_acc: 0.4569
Epoch 12/50
26/26 [==============================] - 4s 158ms/step - loss: 1.8671 - acc: 0.331
9 - val_loss: 1.8951 - val_acc: 0.3495
Epoch 13/50
26/26 [==============================] - 4s 158ms/step - loss: 1.8622 - acc: 0.330
8 - val_loss: 1.4291 - val_acc: 0.5136
Epoch 14/50
26/26 [==============================] - 4s 159ms/step - loss: 1.7936 - acc: 0.353
7 - val_loss: 1.7401 - val_acc: 0.3731
Epoch 15/50
26/26 [==============================] - 4s 159ms/step - loss: 1.7190 - acc: 0.385
0 - val_loss: 1.3079 - val_acc: 0.5077
Epoch 16/50
26/26 [==============================] - 4s 159ms/step - loss: 1.5999 - acc: 0.411
5 - val_loss: 1.2604 - val_acc: 0.5384
Epoch 17/50
26/26 [==============================] - 4s 159ms/step - loss: 1.6304 - acc: 0.415
2 - val_loss: 1.3096 - val_acc: 0.5561
Epoch 18/50
26/26 [==============================] - 4s 159ms/step - loss: 1.5392 - acc: 0.429
8 - val_loss: 1.4380 - val_acc: 0.4817
Epoch 19/50
26/26 [==============================] - 4s 159ms/step - loss: 1.5410 - acc: 0.440
3 - val_loss: 1.1620 - val_acc: 0.5714
Epoch 20/50
26/26 [==============================] - 4s 159ms/step - loss: 1.4697 - acc: 0.470
9 - val_loss: 1.0108 - val_acc: 0.6293
Epoch 21/50
```

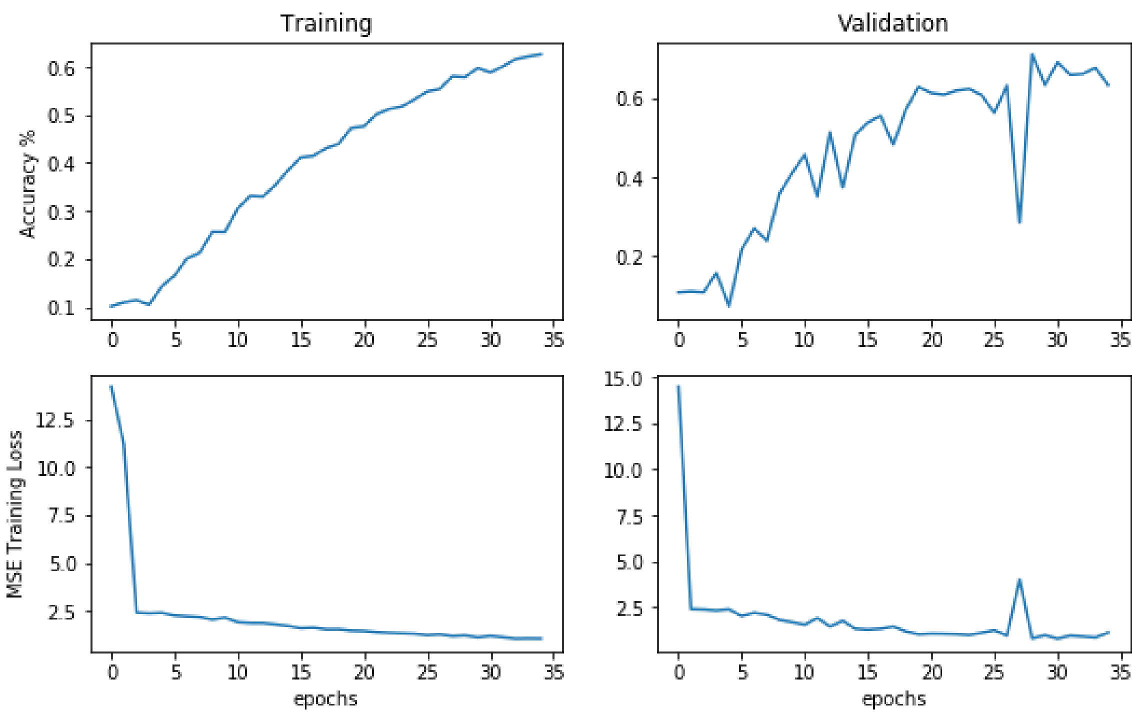
```
26/26 [==============================] - 4s 159ms/step - loss: 1.4533 - acc: 0.476
1 - val_loss: 1.0444 - val_acc: 0.6139
Epoch 22/50
26/26 [==============================] - 4s 159ms/step - loss: 1.3834 - acc: 0.502
2 - val_loss: 1.0368 - val_acc: 0.6092
Epoch 23/50
26/26 [==============================] - 4s 159ms/step - loss: 1.3535 - acc: 0.509
4 - val_loss: 1.0186 - val_acc: 0.6198
Epoch 24/50
26/26 [==============================] - 4s 159ms/step - loss: 1.3319 - acc: 0.517
5 - val_loss: 0.9741 - val_acc: 0.6246
Epoch 25/50
26/26 [==============================] - 4s 159ms/step - loss: 1.3081 - acc: 0.530
8 - val_loss: 1.0872 - val_acc: 0.6080
Epoch 26/50
26/26 [==============================] - 4s 159ms/step - loss: 1.2378 - acc: 0.550
8 - val_loss: 1.2259 - val_acc: 0.5632
Epoch 27/50
26/26 [==============================] - 4s 160ms/step - loss: 1.2732 - acc: 0.556
2 - val_loss: 0.9375 - val_acc: 0.6328
Epoch 28/50
26/26 [==============================] - 4s 162ms/step - loss: 1.1900 - acc: 0.580
5 - val_loss: 3.9850 - val_acc: 0.2834
Epoch 29/50
26/26 [==============================] - 4s 159ms/step - loss: 1.2975 - acc: 0.570
6 - val_loss: 0.7923 - val_acc: 0.7119
Epoch 30/50
26/26 [==============================] - 4s 159ms/step - loss: 1.1245 - acc: 0.597
4 - val_loss: 0.9612 - val_acc: 0.6340
Epoch 31/50
26/26 [==============================] - 4s 159ms/step - loss: 1.2032 - acc: 0.586
5 - val_loss: 0.7779 - val_acc: 0.6919
Epoch 32/50
26/26 [==============================] - 4s 159ms/step - loss: 1.1324 - acc: 0.600
0 - val_loss: 0.9402 - val_acc: 0.6600
Epoch 33/50
26/26 [==============================] - 4s 159ms/step - loss: 1.0586 - acc: 0.618
4 - val_loss: 0.8941 - val_acc: 0.6623
Epoch 34/50
26/26 [==============================] - 4s 159ms/step - loss: 1.0729 - acc: 0.620
5 - val_loss: 0.8401 - val_acc: 0.6777
Epoch 35/50
26/26 [==============================] - 4s 159ms/step - loss: 1.0616 - acc: 0.625
8 - val_loss: 1.0968 - val_acc: 0.6340
```

In [16]:

```
from matplotlib import pyplot as plt

%matplotlib inline

plt.figure(figsize=(10,6))
plt.subplot(2,2,1)
plt.plot(history1.history['acc'])

plt.ylabel('Accuracy %')
plt.title('Training')
plt.subplot(2,2,2)
plt.plot(history1.history['val_acc'])
plt.title('Validation')

plt.subplot(2,2,3)
plt.plot(history1.history['loss'])
plt.ylabel('MSE Training Loss')
plt.xlabel('epochs')

plt.subplot(2,2,4)
plt.plot(history1.history['val_loss'])
plt.xlabel('epochs')
```

Out[16]:

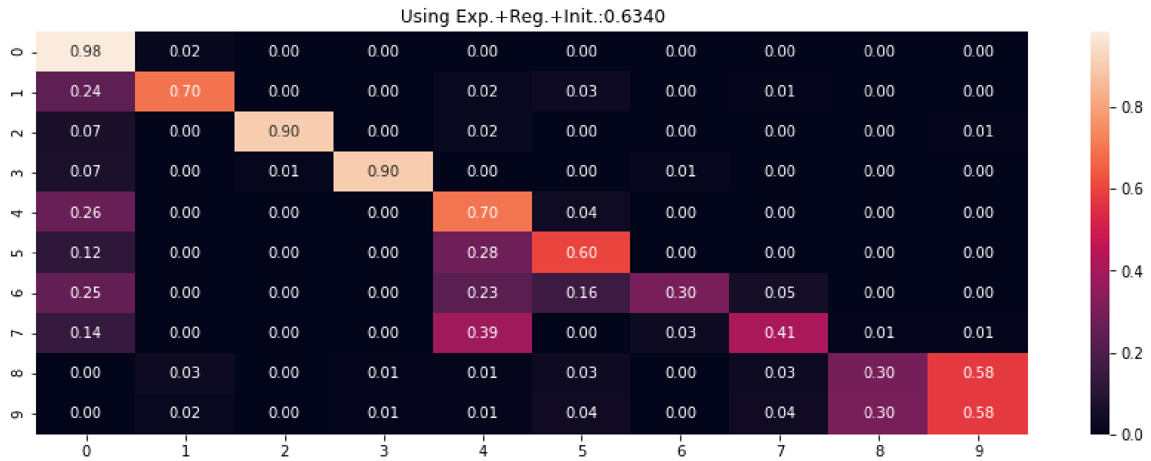
```
Text(0.5, 0, 'epochs')
```



For train vs loss, I think it converge, the line reach the bottom. For validation, the line is boucing at 27 epochs. But in general, they converge.

In [17]:

```
#   0- Apple_Black_rot',
#   1- Apple_healthy',
#   2- Grape_Black_rot',
#   3- Grape_healthy',
#   4- Pepper_Bacterial_spot',
#   5- Pepper_healthy',
#   6- Potato_healthy',
#   7- Potato_Late_blight',
#   8- Tomato_Bacterial_spot',
#   9- Tomato_healthy']

summarize_net(cnn1, X_test, y_test, title_text='Using Exp.+Reg.+Init.:')
```



In [12]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#
sklearn.model_selection.StratifiedKFold.split
#https://github.com/Thakugan/machine-learning-notebooks/blob/master/6-wide-and-deep-networks/mus
hroom-hunting.ipynb
#I used this in last lab, just modified to new version
from sklearn.model_selection import StratifiedKFold
num_folds = 5

acc_scores1 =[]
skf1 = StratifiedKFold(n_splits=num_folds, shuffle=True)
for i, (train, test) in enumerate(skf1.split(X, y)):#here I used corss validation on my model

    cnn1 = create_cnn1()
    #doing modeling same as above, without for loop

    cnn1.fit_generator(datagen.flow(X_train, y_train_ohe, batch_size=128),
                  steps_per_epoch=int(len(X_train)/128), # how many generators to go through per
epoch
                  epochs=50, verbose=1,
                  validation_data=(X_test,y_test_ohe),
                  callbacks=[EarlyStopping(monitor='val_loss', patience=4)]
                 )
#this is just what i do without cross validation
    yhat = np.argmax(cnn1.predict(X_test), axis=1)

    acc_score1 = mt.accuracy_score(y_test, yhat)
    acc_scores1.append(acc_score1)#append all acc for k-fold
    print("Accuracy: ", acc_score1)
print(acc_scores1)
```

```
WARNING:tensorflow:From D:\APP\conda\lib\site-packages\tensorflow\python\ops\math_
ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will
be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/50
26/26 [==============================] - 24s 918ms/step - loss: 3.5369 - acc: 0.10
94 - val_loss: 2.4038 - val_acc: 0.1204
Epoch 2/50
26/26 [==============================] - 5s 197ms/step - loss: 2.5136 - acc: 0.121
0 - val_loss: 2.3286 - val_acc: 0.1617
Epoch 3/50
26/26 [==============================] - 4s 164ms/step - loss: 2.3482 - acc: 0.159
7 - val_loss: 2.3165 - val_acc: 0.1854
Epoch 4/50
26/26 [==============================] - 4s 172ms/step - loss: 2.2404 - acc: 0.205
3 - val_loss: 2.0874 - val_acc: 0.3129
Epoch 5/50
26/26 [==============================] - 4s 159ms/step - loss: 2.2037 - acc: 0.227
8 - val_loss: 1.9945 - val_acc: 0.3140
Epoch 6/50
26/26 [==============================] - 4s 159ms/step - loss: 2.1223 - acc: 0.244
7 - val_loss: 2.4191 - val_acc: 0.1393
Epoch 7/50
26/26 [==============================] - 4s 158ms/step - loss: 2.1450 - acc: 0.266
7 - val_loss: 1.8822 - val_acc: 0.3412
Epoch 8/50
26/26 [==============================] - 4s 158ms/step - loss: 1.9145 - acc: 0.330
2 - val_loss: 1.5090 - val_acc: 0.4687
Epoch 9/50
26/26 [==============================] - 4s 158ms/step - loss: 1.8773 - acc: 0.341
8 - val_loss: 1.4154 - val_acc: 0.4959
Epoch 10/50
26/26 [==============================] - 4s 159ms/step - loss: 1.8502 - acc: 0.359
3 - val_loss: 1.6340 - val_acc: 0.4569
Epoch 11/50
26/26 [==============================] - 4s 159ms/step - loss: 1.7371 - acc: 0.393
7 - val_loss: 1.4276 - val_acc: 0.5159
Epoch 12/50
26/26 [==============================] - 4s 159ms/step - loss: 1.7693 - acc: 0.405
4 - val_loss: 1.3271 - val_acc: 0.5325
Epoch 13/50
26/26 [==============================] - 4s 159ms/step - loss: 1.5835 - acc: 0.441
0 - val_loss: 1.3614 - val_acc: 0.5183
Epoch 14/50
26/26 [==============================] - 4s 160ms/step - loss: 1.4971 - acc: 0.478
1 - val_loss: 1.0272 - val_acc: 0.6222
Epoch 15/50
26/26 [==============================] - 4s 159ms/step - loss: 1.4579 - acc: 0.502
1 - val_loss: 1.2089 - val_acc: 0.5443
Epoch 16/50
26/26 [==============================] - 4s 159ms/step - loss: 1.4224 - acc: 0.524
6 - val_loss: 0.9909 - val_acc: 0.6588
Epoch 17/50
26/26 [==============================] - 4s 159ms/step - loss: 1.3317 - acc: 0.535
8 - val_loss: 1.0578 - val_acc: 0.5986
Epoch 18/50
26/26 [==============================] - 4s 159ms/step - loss: 1.2848 - acc: 0.547
6 - val_loss: 0.9366 - val_acc: 0.6529
Epoch 19/50
26/26 [==============================] - 4s 159ms/step - loss: 1.2464 - acc: 0.570
```

```
9 - val_loss: 0.9312 - val_acc: 0.6375
Epoch 20/50
26/26 [==============================] - 4s 159ms/step - loss: 1.4182 - acc: 0.565
9 - val_loss: 0.8635 - val_acc: 0.7001
Epoch 21/50
26/26 [==============================] - 4s 159ms/step - loss: 1.1409 - acc: 0.611
7 - val_loss: 0.8178 - val_acc: 0.7013
Epoch 22/50
26/26 [==============================] - 4s 159ms/step - loss: 1.1656 - acc: 0.602
8 - val_loss: 0.8256 - val_acc: 0.7143
Epoch 23/50
26/26 [==============================] - 4s 159ms/step - loss: 1.0847 - acc: 0.624
2 - val_loss: 0.7127 - val_acc: 0.7450
Epoch 24/50
26/26 [==============================] - 4s 159ms/step - loss: 1.2113 - acc: 0.599
3 - val_loss: 0.7336 - val_acc: 0.7485
Epoch 25/50
26/26 [==============================] - 4s 159ms/step - loss: 1.0610 - acc: 0.632
0 - val_loss: 0.7667 - val_acc: 0.7344
Epoch 26/50
26/26 [==============================] - 4s 159ms/step - loss: 1.0278 - acc: 0.651
6 - val_loss: 0.8224 - val_acc: 0.7190
Epoch 27/50
26/26 [==============================] - 4s 164ms/step - loss: 1.0370 - acc: 0.655
4 - val_loss: 0.7340 - val_acc: 0.7426
Accuracy:  0.7426210153482881
Epoch 1/50
26/26 [==============================] - 6s 235ms/step - loss: 3.5321 - acc: 0.114
8 - val_loss: 2.4123 - val_acc: 0.1051
Epoch 2/50
26/26 [==============================] - 4s 159ms/step - loss: 2.4013 - acc: 0.111
3 - val_loss: 2.3948 - val_acc: 0.1700
Epoch 3/50
26/26 [==============================] - 4s 159ms/step - loss: 2.4077 - acc: 0.110
7 - val_loss: 2.7915 - val_acc: 0.1169
Epoch 4/50
26/26 [==============================] - 4s 159ms/step - loss: 2.3968 - acc: 0.121
8 - val_loss: 2.3962 - val_acc: 0.1122
Epoch 5/50
26/26 [==============================] - 4s 159ms/step - loss: 2.3560 - acc: 0.115
8 - val_loss: 2.3356 - val_acc: 0.1653
Epoch 6/50
26/26 [==============================] - 4s 161ms/step - loss: 2.3566 - acc: 0.117
6 - val_loss: 2.3328 - val_acc: 0.1370
Epoch 7/50
26/26 [==============================] - 4s 159ms/step - loss: 2.3753 - acc: 0.132
6 - val_loss: 2.2877 - val_acc: 0.1594
Epoch 8/50
26/26 [==============================] - 4s 159ms/step - loss: 2.3004 - acc: 0.142
9 - val_loss: 2.1794 - val_acc: 0.2102
Epoch 9/50
26/26 [==============================] - 4s 158ms/step - loss: 2.3885 - acc: 0.161
4 - val_loss: 2.2497 - val_acc: 0.1771
Epoch 10/50
26/26 [==============================] - 4s 159ms/step - loss: 2.2369 - acc: 0.174
9 - val_loss: 2.1230 - val_acc: 0.2397
Epoch 11/50
26/26 [==============================] - 4s 159ms/step - loss: 2.1802 - acc: 0.198
1 - val_loss: 1.9994 - val_acc: 0.2739
Epoch 12/50
26/26 [==============================] - 4s 159ms/step - loss: 2.2348 - acc: 0.201
```

```
7 - val_loss: 2.0681 - val_acc: 0.2503
Epoch 13/50
26/26 [==============================] - 4s 159ms/step - loss: 2.0574 - acc: 0.229
1 - val_loss: 2.0199 - val_acc: 0.2562
Epoch 14/50
26/26 [==============================] - 4s 159ms/step - loss: 2.1468 - acc: 0.205
5 - val_loss: 1.8417 - val_acc: 0.3318
Epoch 15/50
26/26 [==============================] - 4s 160ms/step - loss: 2.0177 - acc: 0.251
1 - val_loss: 1.9184 - val_acc: 0.2774
Epoch 16/50
26/26 [==============================] - 4s 159ms/step - loss: 1.9832 - acc: 0.283
4 - val_loss: 1.9400 - val_acc: 0.2810
Epoch 17/50
26/26 [==============================] - 4s 159ms/step - loss: 1.9810 - acc: 0.269
7 - val_loss: 1.7371 - val_acc: 0.3967
Epoch 18/50
26/26 [==============================] - 4s 159ms/step - loss: 1.9132 - acc: 0.302
4 - val_loss: 1.7715 - val_acc: 0.4026
Epoch 19/50
26/26 [==============================] - 4s 159ms/step - loss: 1.9094 - acc: 0.300
6 - val_loss: 1.7199 - val_acc: 0.4274
Epoch 20/50
26/26 [==============================] - 4s 159ms/step - loss: 1.9101 - acc: 0.327
6 - val_loss: 1.5608 - val_acc: 0.4664
Epoch 21/50
26/26 [==============================] - 4s 165ms/step - loss: 1.7393 - acc: 0.363
1 - val_loss: 1.5115 - val_acc: 0.4664
Epoch 22/50
26/26 [==============================] - 4s 161ms/step - loss: 1.8019 - acc: 0.357
3 - val_loss: 1.3522 - val_acc: 0.5809
Epoch 23/50
26/26 [==============================] - 4s 162ms/step - loss: 1.6653 - acc: 0.398
7 - val_loss: 1.6376 - val_acc: 0.4557
Epoch 24/50
26/26 [==============================] - 4s 165ms/step - loss: 1.6584 - acc: 0.412
7 - val_loss: 1.3198 - val_acc: 0.5679
Epoch 25/50
26/26 [==============================] - 4s 161ms/step - loss: 1.6165 - acc: 0.413
2 - val_loss: 1.2718 - val_acc: 0.5561
Epoch 26/50
26/26 [==============================] - 4s 163ms/step - loss: 1.6125 - acc: 0.416
0 - val_loss: 1.1622 - val_acc: 0.6033
Epoch 27/50
26/26 [==============================] - 4s 166ms/step - loss: 1.5536 - acc: 0.437
2 - val_loss: 1.1778 - val_acc: 0.6033
Epoch 28/50
26/26 [==============================] - 4s 169ms/step - loss: 1.4599 - acc: 0.466
0 - val_loss: 1.5372 - val_acc: 0.4392
Epoch 29/50
26/26 [==============================] - 4s 165ms/step - loss: 1.4694 - acc: 0.473
6 - val_loss: 1.0056 - val_acc: 0.6482
Epoch 30/50
26/26 [==============================] - 4s 163ms/step - loss: 1.4596 - acc: 0.474
5 - val_loss: 1.0991 - val_acc: 0.6482
Epoch 31/50
26/26 [==============================] - 4s 161ms/step - loss: 1.4215 - acc: 0.503
7 - val_loss: 1.0937 - val_acc: 0.6092
Epoch 32/50
26/26 [==============================] - 4s 171ms/step - loss: 1.3734 - acc: 0.504
1 - val_loss: 1.3477 - val_acc: 0.5502
```

```
Epoch 33/50
26/26 [==============================] - 4s 169ms/step - loss: 1.3905 - acc: 0.495
9 - val_loss: 1.2636 - val_acc: 0.5584
Accuracy:  0.5584415584415584
Epoch 1/50
26/26 [==============================] - 6s 238ms/step - loss: 3.3268 - acc: 0.107
9 - val_loss: 2.4025 - val_acc: 0.1476
Epoch 2/50
26/26 [==============================] - 4s 163ms/step - loss: 2.8366 - acc: 0.136
2 - val_loss: 2.2635 - val_acc: 0.1877
Epoch 3/50
26/26 [==============================] - 4s 159ms/step - loss: 2.3176 - acc: 0.153
0 - val_loss: 2.3600 - val_acc: 0.1370
Epoch 4/50
26/26 [==============================] - 4s 159ms/step - loss: 2.3711 - acc: 0.160
1 - val_loss: 2.2990 - val_acc: 0.1818
Epoch 5/50
26/26 [==============================] - 4s 160ms/step - loss: 2.3211 - acc: 0.177
1 - val_loss: 2.1283 - val_acc: 0.3353
Epoch 6/50
26/26 [==============================] - 4s 160ms/step - loss: 2.4834 - acc: 0.216
2 - val_loss: 1.9720 - val_acc: 0.3542
Epoch 7/50
26/26 [==============================] - 4s 159ms/step - loss: 2.2626 - acc: 0.239
8 - val_loss: 1.8299 - val_acc: 0.3636
Epoch 8/50
26/26 [==============================] - 4s 159ms/step - loss: 2.0214 - acc: 0.287
2 - val_loss: 1.9105 - val_acc: 0.3259
Epoch 9/50
26/26 [==============================] - 4s 161ms/step - loss: 1.9946 - acc: 0.305
7 - val_loss: 1.6946 - val_acc: 0.4168
Epoch 10/50
26/26 [==============================] - 4s 160ms/step - loss: 1.8841 - acc: 0.328
2 - val_loss: 1.5971 - val_acc: 0.4604
Epoch 11/50
26/26 [==============================] - 4s 159ms/step - loss: 1.8366 - acc: 0.359
3 - val_loss: 1.6352 - val_acc: 0.4534
Epoch 12/50
26/26 [==============================] - 4s 160ms/step - loss: 1.7787 - acc: 0.378
0 - val_loss: 1.5072 - val_acc: 0.4994
Epoch 13/50
26/26 [==============================] - 4s 160ms/step - loss: 1.6585 - acc: 0.419
6 - val_loss: 1.4137 - val_acc: 0.4935
Epoch 14/50
26/26 [==============================] - 4s 160ms/step - loss: 1.5734 - acc: 0.444
3 - val_loss: 1.1579 - val_acc: 0.5773
Epoch 15/50
26/26 [==============================] - 4s 160ms/step - loss: 1.5926 - acc: 0.451
7 - val_loss: 1.0256 - val_acc: 0.6505
Epoch 16/50
26/26 [==============================] - 4s 163ms/step - loss: 1.4278 - acc: 0.501
5 - val_loss: 1.1508 - val_acc: 0.5750
Epoch 17/50
26/26 [==============================] - 4s 161ms/step - loss: 1.4229 - acc: 0.511
0 - val_loss: 1.0919 - val_acc: 0.6116
Epoch 18/50
26/26 [==============================] - 4s 161ms/step - loss: 1.4998 - acc: 0.493
2 - val_loss: 1.1681 - val_acc: 0.5950
Epoch 19/50
26/26 [==============================] - 4s 169ms/step - loss: 1.3150 - acc: 0.538
6 - val_loss: 1.2266 - val_acc: 0.5950
```

```
Accuracy:  0.5950413223140496
Epoch 1/50
26/26 [==============================] - 6s 244ms/step - loss: 3.5470 - acc: 0.119
9 - val_loss: 3.6102 - val_acc: 0.1122
Epoch 2/50
26/26 [==============================] - 4s 164ms/step - loss: 3.8797 - acc: 0.112
8 - val_loss: 2.3972 - val_acc: 0.1051
Epoch 3/50
26/26 [==============================] - 4s 164ms/step - loss: 2.4758 - acc: 0.123
4 - val_loss: 2.3854 - val_acc: 0.1051
Epoch 4/50
26/26 [==============================] - 4s 164ms/step - loss: 2.3759 - acc: 0.108
8 - val_loss: 2.2989 - val_acc: 0.1960
Epoch 5/50
26/26 [==============================] - 4s 164ms/step - loss: 2.7490 - acc: 0.132
9 - val_loss: 2.3638 - val_acc: 0.1086
Epoch 6/50
26/26 [==============================] - 4s 166ms/step - loss: 2.2639 - acc: 0.171
5 - val_loss: 2.1568 - val_acc: 0.2255
Epoch 7/50
26/26 [==============================] - 4s 166ms/step - loss: 2.2069 - acc: 0.201
3 - val_loss: 2.0702 - val_acc: 0.2668
Epoch 8/50
26/26 [==============================] - 4s 166ms/step - loss: 2.0955 - acc: 0.246
8 - val_loss: 1.8517 - val_acc: 0.3813
Epoch 9/50
26/26 [==============================] - 4s 165ms/step - loss: 2.3361 - acc: 0.277
9 - val_loss: 2.1169 - val_acc: 0.2385
Epoch 10/50
26/26 [==============================] - 4s 161ms/step - loss: 1.9421 - acc: 0.322
9 - val_loss: 1.7281 - val_acc: 0.4416
Epoch 11/50
26/26 [==============================] - 4s 160ms/step - loss: 1.8460 - acc: 0.347
4 - val_loss: 1.5403 - val_acc: 0.4711
Epoch 12/50
26/26 [==============================] - 4s 159ms/step - loss: 1.7683 - acc: 0.387
3 - val_loss: 1.3549 - val_acc: 0.5384
Epoch 13/50
26/26 [==============================] - 4s 159ms/step - loss: 1.6958 - acc: 0.400
8 - val_loss: 1.2779 - val_acc: 0.5561
Epoch 14/50
26/26 [==============================] - 4s 159ms/step - loss: 1.7524 - acc: 0.409
1 - val_loss: 1.2669 - val_acc: 0.5478
Epoch 15/50
26/26 [==============================] - 4s 159ms/step - loss: 1.5494 - acc: 0.461
1 - val_loss: 1.1503 - val_acc: 0.6305
Epoch 16/50
26/26 [==============================] - 4s 159ms/step - loss: 1.6290 - acc: 0.477
1 - val_loss: 1.2728 - val_acc: 0.5891
Epoch 17/50
26/26 [==============================] - 4s 159ms/step - loss: 1.4291 - acc: 0.515
4 - val_loss: 1.0296 - val_acc: 0.6246
Epoch 18/50
26/26 [==============================] - 4s 160ms/step - loss: 1.4501 - acc: 0.511
9 - val_loss: 1.7551 - val_acc: 0.4321
Epoch 19/50
26/26 [==============================] - 4s 159ms/step - loss: 1.3168 - acc: 0.550
9 - val_loss: 1.3932 - val_acc: 0.4994
Epoch 20/50
26/26 [==============================] - 4s 159ms/step - loss: 1.3091 - acc: 0.549
9 - val_loss: 0.8646 - val_acc: 0.6777
```

```
Epoch 21/50
26/26 [==============================] - 4s 160ms/step - loss: 1.2948 - acc: 0.563
2 - val_loss: 1.3055 - val_acc: 0.5620
Epoch 22/50
26/26 [==============================] - 4s 159ms/step - loss: 1.2209 - acc: 0.586
5 - val_loss: 1.0779 - val_acc: 0.6198
Epoch 23/50
26/26 [==============================] - 4s 159ms/step - loss: 1.2796 - acc: 0.589
5 - val_loss: 0.9156 - val_acc: 0.6494
Epoch 24/50
26/26 [==============================] - 4s 159ms/step - loss: 1.1378 - acc: 0.613
6 - val_loss: 0.9062 - val_acc: 0.6800
Accuracy:  0.680047225501771
Epoch 1/50
26/26 [==============================] - 6s 241ms/step - loss: 3.6050 - acc: 0.108
5 - val_loss: 2.4155 - val_acc: 0.1240
Epoch 2/50
26/26 [==============================] - 4s 160ms/step - loss: 2.4436 - acc: 0.121
8 - val_loss: 2.3978 - val_acc: 0.1263
Epoch 3/50
26/26 [==============================] - 4s 160ms/step - loss: 2.4268 - acc: 0.122
6 - val_loss: 2.3691 - val_acc: 0.1192
Epoch 4/50
26/26 [==============================] - 4s 159ms/step - loss: 2.5666 - acc: 0.121
5 - val_loss: 2.3717 - val_acc: 0.1228
Epoch 5/50
26/26 [==============================] - 4s 160ms/step - loss: 2.3770 - acc: 0.129
1 - val_loss: 2.3621 - val_acc: 0.0933
Epoch 6/50
26/26 [==============================] - 4s 160ms/step - loss: 2.3644 - acc: 0.143
8 - val_loss: 2.3065 - val_acc: 0.1488
Epoch 7/50
26/26 [==============================] - 4s 159ms/step - loss: 2.4599 - acc: 0.140
8 - val_loss: 2.3246 - val_acc: 0.1724
Epoch 8/50
26/26 [==============================] - 4s 159ms/step - loss: 2.2635 - acc: 0.179
7 - val_loss: 2.1894 - val_acc: 0.2574
Epoch 9/50
26/26 [==============================] - 4s 159ms/step - loss: 2.3006 - acc: 0.186
5 - val_loss: 2.1396 - val_acc: 0.2243
Epoch 10/50
26/26 [==============================] - 4s 159ms/step - loss: 2.2020 - acc: 0.221
6 - val_loss: 2.2275 - val_acc: 0.1995
Epoch 11/50
26/26 [==============================] - 4s 160ms/step - loss: 2.1170 - acc: 0.254
5 - val_loss: 2.1979 - val_acc: 0.2798
Epoch 12/50
26/26 [==============================] - 4s 160ms/step - loss: 2.0827 - acc: 0.274
3 - val_loss: 1.8089 - val_acc: 0.3825
Epoch 13/50
26/26 [==============================] - 4s 159ms/step - loss: 2.0823 - acc: 0.286
8 - val_loss: 2.0901 - val_acc: 0.3058
Epoch 14/50
26/26 [==============================] - 4s 159ms/step - loss: 1.9355 - acc: 0.332
1 - val_loss: 1.9249 - val_acc: 0.3601
Epoch 15/50
26/26 [==============================] - 4s 159ms/step - loss: 1.8868 - acc: 0.339
9 - val_loss: 1.5367 - val_acc: 0.4876
Epoch 16/50
26/26 [==============================] - 4s 159ms/step - loss: 1.8334 - acc: 0.375
2 - val_loss: 1.5282 - val_acc: 0.4829
```

```
Epoch 17/50
26/26 [==============================] - 4s 159ms/step - loss: 1.7517 - acc: 0.395
5 - val_loss: 1.4029 - val_acc: 0.5030
Epoch 18/50
26/26 [==============================] - 4s 159ms/step - loss: 1.7067 - acc: 0.398
2 - val_loss: 1.3082 - val_acc: 0.5396
Epoch 19/50
26/26 [==============================] - 4s 161ms/step - loss: 1.6013 - acc: 0.446
2 - val_loss: 1.2556 - val_acc: 0.5431
Epoch 20/50
26/26 [==============================] - 4s 163ms/step - loss: 1.5642 - acc: 0.445
6 - val_loss: 1.4272 - val_acc: 0.5195
Epoch 21/50
26/26 [==============================] - 4s 160ms/step - loss: 1.5335 - acc: 0.466
2 - val_loss: 1.0842 - val_acc: 0.6175
Epoch 22/50
26/26 [==============================] - 4s 160ms/step - loss: 1.4703 - acc: 0.485
5 - val_loss: 1.2331 - val_acc: 0.5738
Epoch 23/50
26/26 [==============================] - 4s 160ms/step - loss: 1.4386 - acc: 0.489
4 - val_loss: 1.0603 - val_acc: 0.6541
Epoch 24/50
26/26 [==============================] - 4s 159ms/step - loss: 1.4414 - acc: 0.492
9 - val_loss: 1.0967 - val_acc: 0.6163
Epoch 25/50
26/26 [==============================] - 4s 159ms/step - loss: 1.4131 - acc: 0.510
1 - val_loss: 1.5453 - val_acc: 0.5573
Epoch 26/50
26/26 [==============================] - 4s 159ms/step - loss: 1.3250 - acc: 0.548
3 - val_loss: 1.1257 - val_acc: 0.5750
Epoch 27/50
26/26 [==============================] - 4s 160ms/step - loss: 1.3138 - acc: 0.528
3 - val_loss: 1.1575 - val_acc: 0.5490
Accuracy:  0.5489964580873672
[0.7426210153482881, 0.5584415584415584, 0.5950413223140496, 0.680047225501771, 0.
5489964580873672]
```

Model1 2X2 kernal size

In [14]:

```
# what if we just want to use the validation data??
from keras.callbacks import EarlyStopping
from keras.regularizers import l2
l2_lambda = 0.0001


def create_cnn2():
    # Use Kaiming He to regularize ReLU layers: https://arxiv.org/pdf/1502.01852.pdf
    # Use Glorot/Bengio for linear/sigmoid/softmax: http://proceedings.mlr.press/v9/glorot10a/gl
orot10a.pdf
    cnn2 = Sequential()

    cnn2.add(Conv2D(filters=32,
                    input_shape = (img_wh,img_wh,1),
                    kernel_size=(2,2),
                    kernel_initializer='he_uniform',
                    kernel_regularizer=l2(l2_lambda),
                    padding='same',
                    activation='relu',
                    data_format="channels_last")) # more compact syntax

    cnn2.add(Conv2D(filters=32,
                kernel_size=(2,2),
                kernel_initializer='he_uniform',
                kernel_regularizer=l2(l2_lambda),
                padding='same',
                activation='relu',data_format="channels_last"))
    cnn2.add(MaxPooling2D(pool_size=(2, 2), data_format="channels_last"))

    cnn2.add(Conv2D(filters=64,
                input_shape = (img_wh,img_wh,1),
                kernel_size=(2,2),
                kernel_initializer='he_uniform',
                kernel_regularizer=l2(l2_lambda),
                padding='same',
                activation='relu',data_format="channels_last")) # more compact syntax

    cnn2.add(Conv2D(filters=64,
                kernel_size=(2,2),
                kernel_initializer='he_uniform',
                kernel_regularizer=l2(l2_lambda),
                padding='same',
                activation='relu'))
    cnn2.add(MaxPooling2D(pool_size=(2, 2), data_format="channels_last"))

    cnn2.add(Conv2D(filters=128,
                input_shape = (img_wh,img_wh,1),
                kernel_size=(2,2),
                kernel_initializer='he_uniform',
                kernel_regularizer=l2(l2_lambda),
                padding='same',
                activation='relu',data_format="channels_last")) # more compact syntax

    cnn2.add(Conv2D(filters=128,
                kernel_size=(2,2),
                kernel_initializer='he_uniform',
                kernel_regularizer=l2(l2_lambda),
                padding='same',
                activation='relu',data_format="channels_last"))
```

```
    # add one layer on flattened output

    cnn2.add(Flatten())
    cnn2.add(Dropout(0.25)) # add some dropout for regularization after conv layers
    cnn2.add(Dense(128,
              activation='relu',
              kernel_initializer='he_uniform',
              kernel_regularizer=l2(l2_lambda)
          ))
    cnn2.add(Dropout(0.5)) # add some dropout for regularization, again!
    cnn2.add(Dense(NUM_CLASSES,
                 activation='softmax',
                 kernel_initializer='glorot_uniform',
                 kernel_regularizer=l2(l2_lambda)
                ))

    cnn2.compile(loss='categorical_crossentropy', # 'categorical_crossentropy' 'mean_squared_err
or'
                 optimizer='rmsprop', # 'adadelta' 'rmsprop'
                 metrics=['accuracy'])
    return cnn2

cnn2 = create_cnn2()
```

In [35]:

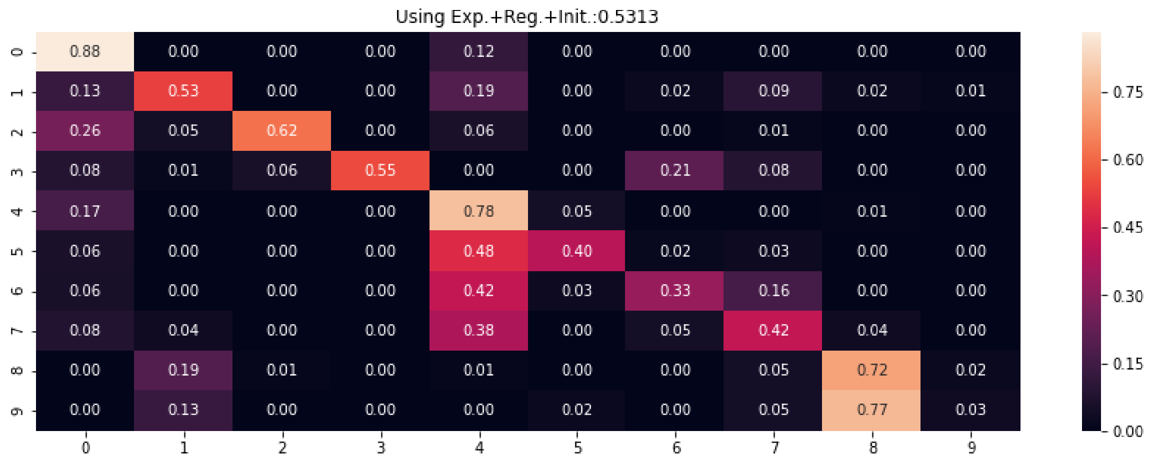
```
# the flow method yields batches of images indefinitely, with the given transofmrations
history2 = cnn2.fit_generator(datagen.flow(X_train, y_train_ohe, batch_size=128),
                  steps_per_epoch=int(len(X_train)/128), # how many generators to go through per
epoch
                  epochs=50, verbose=1,
                  validation_data=(X_test,y_test_ohe),
                  callbacks=[EarlyStopping(monitor='val_loss', patience=4)]
                 )
```

```
Epoch 1/50
26/26 [==============================] - 6s 238ms/step - loss: 3.7957 - acc: 0.122
0 - val_loss: 2.4090 - val_acc: 0.1901
Epoch 2/50
26/26 [==============================] - 4s 143ms/step - loss: 2.4863 - acc: 0.127
3 - val_loss: 2.3890 - val_acc: 0.1818
Epoch 3/50
26/26 [==============================] - 4s 147ms/step - loss: 2.3959 - acc: 0.126
0 - val_loss: 2.3804 - val_acc: 0.1086
Epoch 4/50
26/26 [==============================] - 4s 154ms/step - loss: 2.3990 - acc: 0.140
0 - val_loss: 2.3044 - val_acc: 0.1558
Epoch 5/50
26/26 [==============================] - 4s 154ms/step - loss: 2.2585 - acc: 0.187
5 - val_loss: 2.0656 - val_acc: 0.2267
Epoch 6/50
26/26 [==============================] - 4s 153ms/step - loss: 2.2552 - acc: 0.205
6 - val_loss: 2.0941 - val_acc: 0.3294
Epoch 7/50
26/26 [==============================] - 4s 153ms/step - loss: 2.1381 - acc: 0.221
1 - val_loss: 2.0019 - val_acc: 0.3377
Epoch 8/50
26/26 [==============================] - 4s 154ms/step - loss: 2.1064 - acc: 0.248
5 - val_loss: 1.8532 - val_acc: 0.3731
Epoch 9/50
26/26 [==============================] - 4s 154ms/step - loss: 2.0446 - acc: 0.273
7 - val_loss: 1.8568 - val_acc: 0.3601
Epoch 10/50
26/26 [==============================] - 4s 155ms/step - loss: 1.9767 - acc: 0.290
7 - val_loss: 1.6980 - val_acc: 0.3955
Epoch 11/50
26/26 [==============================] - 4s 154ms/step - loss: 1.8850 - acc: 0.317
9 - val_loss: 1.5758 - val_acc: 0.4451
Epoch 12/50
26/26 [==============================] - 4s 155ms/step - loss: 1.9351 - acc: 0.322
0 - val_loss: 1.4897 - val_acc: 0.5289
Epoch 13/50
26/26 [==============================] - 4s 155ms/step - loss: 1.7888 - acc: 0.377
1 - val_loss: 2.8390 - val_acc: 0.2432
Epoch 14/50
26/26 [==============================] - 4s 155ms/step - loss: 1.8090 - acc: 0.378
8 - val_loss: 1.3718 - val_acc: 0.5053
Epoch 15/50
26/26 [==============================] - 4s 155ms/step - loss: 1.7305 - acc: 0.393
1 - val_loss: 1.4748 - val_acc: 0.4723
Epoch 16/50
26/26 [==============================] - 4s 155ms/step - loss: 1.6338 - acc: 0.424
5 - val_loss: 1.2273 - val_acc: 0.5785
Epoch 17/50
26/26 [==============================] - 4s 155ms/step - loss: 1.5496 - acc: 0.460
6 - val_loss: 1.3465 - val_acc: 0.5077
Epoch 18/50
26/26 [==============================] - 4s 154ms/step - loss: 1.5244 - acc: 0.465
4 - val_loss: 1.1072 - val_acc: 0.6033
Epoch 19/50
26/26 [==============================] - 4s 154ms/step - loss: 1.4979 - acc: 0.490
2 - val_loss: 1.1970 - val_acc: 0.6222
Epoch 20/50
26/26 [==============================] - 4s 154ms/step - loss: 1.4563 - acc: 0.487
6 - val_loss: 1.0484 - val_acc: 0.6364
Epoch 21/50
```

```
26/26 [==============================] - 4s 154ms/step - loss: 1.4268 - acc: 0.509
2 - val_loss: 0.9720 - val_acc: 0.6647
Epoch 22/50
26/26 [==============================] - 4s 155ms/step - loss: 1.3791 - acc: 0.520
4 - val_loss: 1.0825 - val_acc: 0.6080
Epoch 23/50
26/26 [==============================] - 4s 152ms/step - loss: 1.3889 - acc: 0.528
4 - val_loss: 1.0204 - val_acc: 0.6293
Epoch 24/50
26/26 [==============================] - 4s 154ms/step - loss: 1.2991 - acc: 0.547
2 - val_loss: 1.1081 - val_acc: 0.5986
Epoch 25/50
26/26 [==============================] - 4s 153ms/step - loss: 1.2710 - acc: 0.548
2 - val_loss: 1.4816 - val_acc: 0.5313
```

In [36]:

```
#   0-  Apple_Black_rot',
#   1-  Apple_healthy',
#   2-  Grape_Black_rot',
#   3-  Grape_healthy',
#   4-  Pepper_Bacterial_spot',
#   5-  Pepper_healthy',
#   6-  Potato_healthy',
#   7-  Potato_Late_blight',
#   8-  Tomato_Bacterial_spot',
#   9-  Tomato_healthy']

summarize_net(cnn2, X_test, y_test, title_text='Using Exp.+Reg.+Init.:')
```



In [15]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#
sklearn.model_selection.StratifiedKFold.split
#https://github.com/Thakugan/machine-learning-notebooks/blob/master/6-wide-and-deep-networks/mus
hroom-hunting.ipynb
#I used this in last lab, just modified to new version
from sklearn.model_selection import StratifiedKFold
num_folds = 5

acc_scores2 =[]
skf2 = StratifiedKFold(n_splits=num_folds, shuffle=True)
for i, (train, test) in enumerate(skf2.split(X, y)):#here I used corss validation on my model

    cnn2 = create_cnn2()
    #doing modeling same as above, without for loop

    cnn2.fit_generator(datagen.flow(X_train, y_train_ohe, batch_size=128),
                  steps_per_epoch=int(len(X_train)/128), # how many generators to go through per
epoch
                  epochs=50, verbose=1,
                  validation_data=(X_test,y_test_ohe),
                  callbacks=[EarlyStopping(monitor='val_loss', patience=4)]
                 )
#this is just what i do without cross validation
    yhat = np.argmax(cnn2.predict(X_test), axis=1)

    acc_score2 = mt.accuracy_score(y_test, yhat)
    acc_scores2.append(acc_score2)#append all acc for k-fold
    print("Accuracy: ", acc_score2)
print(acc_scores2)
```

```
Epoch 1/50
26/26 [==============================] - 8s 305ms/step - loss: 3.2950 - acc: 0.107
0 - val_loss: 2.3881 - val_acc: 0.1653
Epoch 2/50
26/26 [==============================] - 5s 181ms/step - loss: 2.4534 - acc: 0.131
5 - val_loss: 2.3793 - val_acc: 0.1311
Epoch 3/50
26/26 [==============================] - 4s 154ms/step - loss: 2.3152 - acc: 0.170
3 - val_loss: 2.1715 - val_acc: 0.2302
Epoch 4/50
26/26 [==============================] - 4s 154ms/step - loss: 2.2832 - acc: 0.188
9 - val_loss: 2.2627 - val_acc: 0.2220
Epoch 5/50
26/26 [==============================] - 4s 154ms/step - loss: 2.1733 - acc: 0.226
3 - val_loss: 1.9984 - val_acc: 0.2975
Epoch 6/50
26/26 [==============================] - 4s 154ms/step - loss: 2.0949 - acc: 0.247
5 - val_loss: 2.1277 - val_acc: 0.2834
Epoch 7/50
26/26 [==============================] - 4s 154ms/step - loss: 2.0311 - acc: 0.282
0 - val_loss: 1.9339 - val_acc: 0.3436
Epoch 8/50
26/26 [==============================] - 4s 154ms/step - loss: 2.0254 - acc: 0.286
7 - val_loss: 3.5192 - val_acc: 0.1429
Epoch 9/50
26/26 [==============================] - 4s 154ms/step - loss: 1.9930 - acc: 0.309
8 - val_loss: 1.5396 - val_acc: 0.4522
Epoch 10/50
26/26 [==============================] - 4s 154ms/step - loss: 1.7821 - acc: 0.387
3 - val_loss: 1.9432 - val_acc: 0.3483
Epoch 11/50
26/26 [==============================] - 4s 154ms/step - loss: 1.8145 - acc: 0.371
5 - val_loss: 1.7539 - val_acc: 0.3766
Epoch 12/50
26/26 [==============================] - 4s 154ms/step - loss: 1.6772 - acc: 0.426
3 - val_loss: 1.4339 - val_acc: 0.5041
Epoch 13/50
26/26 [==============================] - 4s 155ms/step - loss: 1.6900 - acc: 0.429
8 - val_loss: 1.4343 - val_acc: 0.5053
Epoch 14/50
26/26 [==============================] - 4s 154ms/step - loss: 1.5111 - acc: 0.464
3 - val_loss: 1.2422 - val_acc: 0.5856
Epoch 15/50
26/26 [==============================] - 4s 154ms/step - loss: 1.4784 - acc: 0.505
9 - val_loss: 1.2386 - val_acc: 0.5537
Epoch 16/50
26/26 [==============================] - 4s 156ms/step - loss: 1.4145 - acc: 0.520
9 - val_loss: 1.0945 - val_acc: 0.6411
Epoch 17/50
26/26 [==============================] - 4s 154ms/step - loss: 1.3973 - acc: 0.517
8 - val_loss: 1.1461 - val_acc: 0.5939
Epoch 18/50
26/26 [==============================] - 4s 154ms/step - loss: 1.4406 - acc: 0.523
7 - val_loss: 1.1858 - val_acc: 0.5844
Epoch 19/50
26/26 [==============================] - 4s 154ms/step - loss: 1.2714 - acc: 0.572
5 - val_loss: 1.0905 - val_acc: 0.5986
Epoch 20/50
26/26 [==============================] - 4s 154ms/step - loss: 1.2933 - acc: 0.555
5 - val_loss: 1.1880 - val_acc: 0.5868
Epoch 21/50
```

```
26/26 [==============================] - 4s 154ms/step - loss: 1.2984 - acc: 0.571
8 - val_loss: 0.9570 - val_acc: 0.6553
Epoch 22/50
26/26 [==============================] - 4s 155ms/step - loss: 1.2032 - acc: 0.611
3 - val_loss: 0.9474 - val_acc: 0.6470
Epoch 23/50
26/26 [==============================] - 4s 153ms/step - loss: 1.2085 - acc: 0.590
1 - val_loss: 1.2870 - val_acc: 0.5325
Epoch 24/50
26/26 [==============================] - 4s 154ms/step - loss: 1.2128 - acc: 0.591
5 - val_loss: 1.3923 - val_acc: 0.5502
Epoch 25/50
26/26 [==============================] - 4s 154ms/step - loss: 1.1473 - acc: 0.613
8 - val_loss: 1.0136 - val_acc: 0.6387
Epoch 26/50
26/26 [==============================] - 4s 155ms/step - loss: 1.1591 - acc: 0.606
9 - val_loss: 0.9192 - val_acc: 0.6682
Epoch 27/50
26/26 [==============================] - 4s 159ms/step - loss: 1.0961 - acc: 0.633
3 - val_loss: 1.3742 - val_acc: 0.5277
Epoch 28/50
26/26 [==============================] - 4s 157ms/step - loss: 1.1148 - acc: 0.625
6 - val_loss: 1.0034 - val_acc: 0.6529
Epoch 29/50
26/26 [==============================] - 4s 154ms/step - loss: 1.0994 - acc: 0.632
2 - val_loss: 1.2136 - val_acc: 0.5762
Epoch 30/50
26/26 [==============================] - 4s 154ms/step - loss: 1.0961 - acc: 0.632
8 - val_loss: 0.9318 - val_acc: 0.6682
Accuracy:  0.6682408500590319
Epoch 1/50
26/26 [==============================] - 6s 242ms/step - loss: 4.1171 - acc: 0.101
0 - val_loss: 2.3658 - val_acc: 0.1358
Epoch 2/50
26/26 [==============================] - 4s 154ms/step - loss: 2.4263 - acc: 0.139
5 - val_loss: 2.3563 - val_acc: 0.1558
Epoch 3/50
26/26 [==============================] - 4s 154ms/step - loss: 2.3518 - acc: 0.162
1 - val_loss: 2.1681 - val_acc: 0.2609
Epoch 4/50
26/26 [==============================] - 4s 154ms/step - loss: 2.3124 - acc: 0.198
7 - val_loss: 2.0384 - val_acc: 0.2975
Epoch 5/50
26/26 [==============================] - 4s 153ms/step - loss: 2.2534 - acc: 0.217
5 - val_loss: 1.9327 - val_acc: 0.3388
Epoch 6/50
26/26 [==============================] - 4s 153ms/step - loss: 2.1012 - acc: 0.248
1 - val_loss: 1.9794 - val_acc: 0.3282
Epoch 7/50
26/26 [==============================] - 4s 153ms/step - loss: 2.1109 - acc: 0.263
5 - val_loss: 1.8124 - val_acc: 0.3719
Epoch 8/50
26/26 [==============================] - 4s 154ms/step - loss: 1.9806 - acc: 0.301
9 - val_loss: 1.8849 - val_acc: 0.3625
Epoch 9/50
26/26 [==============================] - 4s 155ms/step - loss: 1.9307 - acc: 0.333
4 - val_loss: 1.7198 - val_acc: 0.4191
Epoch 10/50
26/26 [==============================] - 4s 155ms/step - loss: 1.8307 - acc: 0.365
1 - val_loss: 1.5297 - val_acc: 0.5159
Epoch 11/50
```

```
26/26 [==============================] - 4s 154ms/step - loss: 1.7042 - acc: 0.408
0 - val_loss: 1.3960 - val_acc: 0.5396
Epoch 12/50
26/26 [==============================] - 4s 155ms/step - loss: 1.7545 - acc: 0.423
6 - val_loss: 1.3046 - val_acc: 0.5525
Epoch 13/50
26/26 [==============================] - 4s 155ms/step - loss: 1.5898 - acc: 0.468
1 - val_loss: 1.3325 - val_acc: 0.5396
Epoch 14/50
26/26 [==============================] - 4s 154ms/step - loss: 1.5323 - acc: 0.478
1 - val_loss: 1.4048 - val_acc: 0.5514
Epoch 15/50
26/26 [==============================] - 4s 154ms/step - loss: 1.4383 - acc: 0.510
4 - val_loss: 1.1227 - val_acc: 0.6128
Epoch 16/50
26/26 [==============================] - 4s 154ms/step - loss: 1.4723 - acc: 0.517
4 - val_loss: 1.3046 - val_acc: 0.5525
Epoch 17/50
26/26 [==============================] - 4s 154ms/step - loss: 1.3408 - acc: 0.545
1 - val_loss: 1.0517 - val_acc: 0.6104
Epoch 18/50
26/26 [==============================] - 4s 154ms/step - loss: 1.3667 - acc: 0.543
4 - val_loss: 0.9987 - val_acc: 0.6659
Epoch 19/50
26/26 [==============================] - 4s 154ms/step - loss: 1.2535 - acc: 0.579
9 - val_loss: 0.9649 - val_acc: 0.6682
Epoch 20/50
26/26 [==============================] - 4s 154ms/step - loss: 1.3329 - acc: 0.558
7 - val_loss: 1.1228 - val_acc: 0.5950
Epoch 21/50
26/26 [==============================] - 4s 154ms/step - loss: 1.2167 - acc: 0.591
8 - val_loss: 1.0105 - val_acc: 0.6411
Epoch 22/50
26/26 [==============================] - 4s 154ms/step - loss: 1.1530 - acc: 0.614
9 - val_loss: 0.8665 - val_acc: 0.7107
Epoch 23/50
26/26 [==============================] - 4s 155ms/step - loss: 1.2554 - acc: 0.601
3 - val_loss: 1.1533 - val_acc: 0.6068
Epoch 24/50
26/26 [==============================] - 4s 154ms/step - loss: 1.1279 - acc: 0.621
7 - val_loss: 1.0209 - val_acc: 0.6198
Epoch 25/50
26/26 [==============================] - 4s 154ms/step - loss: 1.1201 - acc: 0.634
0 - val_loss: 0.9110 - val_acc: 0.6836
Epoch 26/50
26/26 [==============================] - 4s 154ms/step - loss: 1.1642 - acc: 0.618
3 - val_loss: 0.8134 - val_acc: 0.7060
Epoch 27/50
26/26 [==============================] - 4s 155ms/step - loss: 1.0621 - acc: 0.643
9 - val_loss: 1.0341 - val_acc: 0.6434
Epoch 28/50
26/26 [==============================] - 4s 156ms/step - loss: 1.2043 - acc: 0.624
7 - val_loss: 0.8648 - val_acc: 0.7226
Epoch 29/50
26/26 [==============================] - 4s 154ms/step - loss: 1.0238 - acc: 0.660
0 - val_loss: 0.9581 - val_acc: 0.6871
Epoch 30/50
26/26 [==============================] - 4s 154ms/step - loss: 1.0528 - acc: 0.652
1 - val_loss: 0.7839 - val_acc: 0.7143
Epoch 31/50
26/26 [==============================] - 4s 154ms/step - loss: 0.9765 - acc: 0.684
```

```
7 - val_loss: 1.1614 - val_acc: 0.6246
Epoch 32/50
26/26 [==============================] - 4s 155ms/step - loss: 1.0072 - acc: 0.672
9 - val_loss: 1.0418 - val_acc: 0.6576
Epoch 33/50
26/26 [==============================] - 4s 154ms/step - loss: 1.0124 - acc: 0.669
5 - val_loss: 0.8015 - val_acc: 0.7308
Epoch 34/50
26/26 [==============================] - 4s 154ms/step - loss: 0.9834 - acc: 0.675
8 - val_loss: 1.0356 - val_acc: 0.6234
Accuracy:  0.6233766233766234
Epoch 1/50
26/26 [==============================] - 6s 247ms/step - loss: 3.7051 - acc: 0.125
6 - val_loss: 2.3884 - val_acc: 0.1110
Epoch 2/50
26/26 [==============================] - 4s 154ms/step - loss: 2.4111 - acc: 0.131
5 - val_loss: 2.3254 - val_acc: 0.1606
Epoch 3/50
26/26 [==============================] - 4s 154ms/step - loss: 2.3430 - acc: 0.156
2 - val_loss: 2.1202 - val_acc: 0.2609
Epoch 4/50
26/26 [==============================] - 4s 154ms/step - loss: 2.2796 - acc: 0.192
3 - val_loss: 2.2409 - val_acc: 0.2149
Epoch 5/50
26/26 [==============================] - 4s 154ms/step - loss: 2.2189 - acc: 0.210
7 - val_loss: 1.9739 - val_acc: 0.2975
Epoch 6/50
26/26 [==============================] - 4s 154ms/step - loss: 2.1286 - acc: 0.231
4 - val_loss: 1.9398 - val_acc: 0.3554
Epoch 7/50
26/26 [==============================] - 4s 154ms/step - loss: 2.0649 - acc: 0.275
1 - val_loss: 2.0379 - val_acc: 0.3105
Epoch 8/50
26/26 [==============================] - 4s 154ms/step - loss: 2.0764 - acc: 0.286
4 - val_loss: 1.9384 - val_acc: 0.3377
Epoch 9/50
26/26 [==============================] - 4s 154ms/step - loss: 1.9563 - acc: 0.324
1 - val_loss: 1.6222 - val_acc: 0.4687
Epoch 10/50
26/26 [==============================] - 4s 153ms/step - loss: 1.8535 - acc: 0.352
1 - val_loss: 1.7957 - val_acc: 0.3613
Epoch 11/50
26/26 [==============================] - 4s 154ms/step - loss: 1.7725 - acc: 0.394
1 - val_loss: 1.3902 - val_acc: 0.5183
Epoch 12/50
26/26 [==============================] - 4s 154ms/step - loss: 1.6827 - acc: 0.413
7 - val_loss: 1.4235 - val_acc: 0.4935
Epoch 13/50
26/26 [==============================] - 4s 154ms/step - loss: 1.6678 - acc: 0.423
9 - val_loss: 1.2803 - val_acc: 0.5502
Epoch 14/50
26/26 [==============================] - 4s 157ms/step - loss: 1.4938 - acc: 0.481
1 - val_loss: 1.3616 - val_acc: 0.5195
Epoch 15/50
26/26 [==============================] - 4s 155ms/step - loss: 1.4908 - acc: 0.487
7 - val_loss: 1.3102 - val_acc: 0.5608
Epoch 16/50
26/26 [==============================] - 4s 154ms/step - loss: 1.4115 - acc: 0.513
7 - val_loss: 1.1062 - val_acc: 0.6021
Epoch 17/50
26/26 [==============================] - 4s 153ms/step - loss: 1.3463 - acc: 0.530
```

```
4 - val_loss: 1.0441 - val_acc: 0.6293
Epoch 18/50
26/26 [==============================] - 4s 154ms/step - loss: 1.6535 - acc: 0.516
7 - val_loss: 1.0610 - val_acc: 0.6116
Epoch 19/50
26/26 [==============================] - 4s 154ms/step - loss: 1.2627 - acc: 0.556
0 - val_loss: 1.0543 - val_acc: 0.6210
Epoch 20/50
26/26 [==============================] - 4s 156ms/step - loss: 1.2477 - acc: 0.572
1 - val_loss: 0.9631 - val_acc: 0.6765
Epoch 21/50
26/26 [==============================] - 4s 158ms/step - loss: 1.2291 - acc: 0.582
9 - val_loss: 0.9872 - val_acc: 0.6588
Epoch 22/50
26/26 [==============================] - 4s 155ms/step - loss: 1.1981 - acc: 0.593
7 - val_loss: 1.0629 - val_acc: 0.6104
Epoch 23/50
26/26 [==============================] - 4s 156ms/step - loss: 1.2205 - acc: 0.575
5 - val_loss: 0.9501 - val_acc: 0.6553
Epoch 24/50
26/26 [==============================] - 4s 156ms/step - loss: 1.1053 - acc: 0.618
9 - val_loss: 1.2129 - val_acc: 0.5844
Epoch 25/50
26/26 [==============================] - 4s 152ms/step - loss: 1.3604 - acc: 0.588
9 - val_loss: 1.0205 - val_acc: 0.6293
Epoch 26/50
26/26 [==============================] - 4s 146ms/step - loss: 1.0507 - acc: 0.641
8 - val_loss: 1.8113 - val_acc: 0.3979
Epoch 27/50
26/26 [==============================] - 4s 155ms/step - loss: 1.0911 - acc: 0.631
8 - val_loss: 0.7968 - val_acc: 0.7072
Epoch 28/50
26/26 [==============================] - 4s 164ms/step - loss: 1.1168 - acc: 0.624
4 - val_loss: 1.1727 - val_acc: 0.5856
Epoch 29/50
26/26 [==============================] - 4s 156ms/step - loss: 1.0650 - acc: 0.638
3 - val_loss: 0.8425 - val_acc: 0.6848
Epoch 30/50
26/26 [==============================] - 4s 156ms/step - loss: 1.0196 - acc: 0.660
6 - val_loss: 1.3244 - val_acc: 0.5821
Epoch 31/50
26/26 [==============================] - 4s 157ms/step - loss: 1.0751 - acc: 0.648
6 - val_loss: 0.8060 - val_acc: 0.7025
Accuracy:  0.7024793388429752
Epoch 1/50
26/26 [==============================] - 7s 256ms/step - loss: 4.1116 - acc: 0.115
1 - val_loss: 2.3983 - val_acc: 0.1098
Epoch 2/50
26/26 [==============================] - 4s 168ms/step - loss: 2.4179 - acc: 0.138
5 - val_loss: 2.3492 - val_acc: 0.1594
Epoch 3/50
26/26 [==============================] - 4s 168ms/step - loss: 2.4001 - acc: 0.154
4 - val_loss: 2.2336 - val_acc: 0.1346
Epoch 4/50
26/26 [==============================] - 4s 156ms/step - loss: 2.2224 - acc: 0.200
6 - val_loss: 2.1271 - val_acc: 0.3188
Epoch 5/50
26/26 [==============================] - 4s 159ms/step - loss: 2.1634 - acc: 0.223
7 - val_loss: 1.9995 - val_acc: 0.3589
Epoch 6/50
26/26 [==============================] - 4s 165ms/step - loss: 2.1274 - acc: 0.247
```

```
8 - val_loss: 1.9449 - val_acc: 0.3306
Epoch 7/50
26/26 [==============================] - 4s 156ms/step - loss: 2.0108 - acc: 0.301
3 - val_loss: 1.7666 - val_acc: 0.3967
Epoch 8/50
26/26 [==============================] - 4s 155ms/step - loss: 2.0477 - acc: 0.302
9 - val_loss: 1.7227 - val_acc: 0.4179
Epoch 9/50
26/26 [==============================] - 4s 155ms/step - loss: 1.8991 - acc: 0.360
0 - val_loss: 1.7535 - val_acc: 0.4168
Epoch 10/50
26/26 [==============================] - 4s 154ms/step - loss: 1.8657 - acc: 0.364
5 - val_loss: 1.8166 - val_acc: 0.3825
Epoch 11/50
26/26 [==============================] - 4s 154ms/step - loss: 1.7981 - acc: 0.402
2 - val_loss: 1.3882 - val_acc: 0.5336
Epoch 12/50
26/26 [==============================] - 4s 155ms/step - loss: 1.6344 - acc: 0.453
8 - val_loss: 1.2910 - val_acc: 0.5396
Epoch 13/50
26/26 [==============================] - 4s 154ms/step - loss: 1.5530 - acc: 0.469
6 - val_loss: 1.3781 - val_acc: 0.5419
Epoch 14/50
26/26 [==============================] - 4s 153ms/step - loss: 1.5986 - acc: 0.474
3 - val_loss: 1.1127 - val_acc: 0.6210
Epoch 15/50
26/26 [==============================] - 4s 155ms/step - loss: 1.3895 - acc: 0.518
5 - val_loss: 1.1828 - val_acc: 0.6128
Epoch 16/50
26/26 [==============================] - 4s 155ms/step - loss: 1.4073 - acc: 0.532
6 - val_loss: 1.1495 - val_acc: 0.5856
Epoch 17/50
26/26 [==============================] - 4s 159ms/step - loss: 1.3632 - acc: 0.537
4 - val_loss: 1.1041 - val_acc: 0.6139
Epoch 18/50
26/26 [==============================] - 4s 154ms/step - loss: 1.3077 - acc: 0.563
1 - val_loss: 1.2224 - val_acc: 0.5478
Epoch 19/50
26/26 [==============================] - 4s 154ms/step - loss: 1.2663 - acc: 0.573
3 - val_loss: 0.9421 - val_acc: 0.6777
Epoch 20/50
26/26 [==============================] - 4s 155ms/step - loss: 1.3975 - acc: 0.562
9 - val_loss: 1.7861 - val_acc: 0.4616
Epoch 21/50
26/26 [==============================] - 4s 155ms/step - loss: 1.1645 - acc: 0.594
8 - val_loss: 0.9603 - val_acc: 0.6588
Epoch 22/50
26/26 [==============================] - 4s 155ms/step - loss: 1.1860 - acc: 0.613
9 - val_loss: 0.8540 - val_acc: 0.7249
Epoch 23/50
26/26 [==============================] - 4s 157ms/step - loss: 1.0976 - acc: 0.629
3 - val_loss: 1.1589 - val_acc: 0.6021
Epoch 24/50
26/26 [==============================] - 4s 159ms/step - loss: 1.1315 - acc: 0.610
9 - val_loss: 0.8859 - val_acc: 0.6812
Epoch 25/50
26/26 [==============================] - 4s 156ms/step - loss: 1.1143 - acc: 0.632
3 - val_loss: 0.8600 - val_acc: 0.6966
Epoch 26/50
26/26 [==============================] - 4s 154ms/step - loss: 1.0994 - acc: 0.641
3 - val_loss: 1.4786 - val_acc: 0.5136
```

```
Accuracy:  0.51357733175915
Epoch 1/50
26/26 [==============================] - 6s 249ms/step - loss: 4.8336 - acc: 0.113
6 - val_loss: 2.4382 - val_acc: 0.1204
Epoch 2/50
26/26 [==============================] - 4s 155ms/step - loss: 2.4942 - acc: 0.148
7 - val_loss: 2.3258 - val_acc: 0.1783
Epoch 3/50
26/26 [==============================] - 4s 155ms/step - loss: 2.3254 - acc: 0.177
0 - val_loss: 2.1573 - val_acc: 0.2798
Epoch 4/50
26/26 [==============================] - 4s 154ms/step - loss: 2.2478 - acc: 0.209
7 - val_loss: 2.4525 - val_acc: 0.1936
Epoch 5/50
26/26 [==============================] - 4s 159ms/step - loss: 2.1674 - acc: 0.236
5 - val_loss: 1.9301 - val_acc: 0.3329
Epoch 6/50
26/26 [==============================] - 4s 156ms/step - loss: 2.1097 - acc: 0.266
7 - val_loss: 2.1196 - val_acc: 0.2562
Epoch 7/50
26/26 [==============================] - 4s 155ms/step - loss: 2.0955 - acc: 0.276
8 - val_loss: 1.8396 - val_acc: 0.3849
Epoch 8/50
26/26 [==============================] - 4s 162ms/step - loss: 1.9541 - acc: 0.331
4 - val_loss: 1.7840 - val_acc: 0.4050
Epoch 9/50
26/26 [==============================] - 4s 154ms/step - loss: 1.9164 - acc: 0.340
2 - val_loss: 1.5484 - val_acc: 0.4994
Epoch 10/50
26/26 [==============================] - 4s 157ms/step - loss: 1.8480 - acc: 0.357
6 - val_loss: 1.7274 - val_acc: 0.4227
Epoch 11/50
26/26 [==============================] - 4s 156ms/step - loss: 1.8173 - acc: 0.403
0 - val_loss: 1.4248 - val_acc: 0.5018
Epoch 12/50
26/26 [==============================] - 4s 155ms/step - loss: 1.7355 - acc: 0.425
8 - val_loss: 1.2817 - val_acc: 0.5762
Epoch 13/50
26/26 [==============================] - 4s 156ms/step - loss: 1.5983 - acc: 0.455
3 - val_loss: 1.2502 - val_acc: 0.5903
Epoch 14/50
26/26 [==============================] - 4s 156ms/step - loss: 1.5611 - acc: 0.475
2 - val_loss: 1.2968 - val_acc: 0.5419
Epoch 15/50
26/26 [==============================] - 4s 156ms/step - loss: 1.4434 - acc: 0.509
9 - val_loss: 1.1129 - val_acc: 0.6222
Epoch 16/50
26/26 [==============================] - 4s 156ms/step - loss: 1.3755 - acc: 0.527
3 - val_loss: 1.1278 - val_acc: 0.5915
Epoch 17/50
26/26 [==============================] - 4s 155ms/step - loss: 1.4191 - acc: 0.534
3 - val_loss: 1.2382 - val_acc: 0.5596
Epoch 18/50
26/26 [==============================] - 4s 156ms/step - loss: 1.3553 - acc: 0.533
1 - val_loss: 1.0670 - val_acc: 0.6198
Epoch 19/50
26/26 [==============================] - 4s 156ms/step - loss: 1.2811 - acc: 0.561
2 - val_loss: 1.0426 - val_acc: 0.6293
Epoch 20/50
26/26 [==============================] - 4s 157ms/step - loss: 1.3528 - acc: 0.547
8 - val_loss: 0.9437 - val_acc: 0.6269
```

```
Epoch 21/50
26/26 [==============================] - 4s 156ms/step - loss: 1.3011 - acc: 0.553
4 - val_loss: 1.4435 - val_acc: 0.5112
Epoch 22/50
26/26 [==============================] - 4s 156ms/step - loss: 1.2648 - acc: 0.558
9 - val_loss: 1.1031 - val_acc: 0.6281
Epoch 23/50
26/26 [==============================] - 4s 156ms/step - loss: 1.2361 - acc: 0.589
1 - val_loss: 0.9556 - val_acc: 0.6824
Epoch 24/50
26/26 [==============================] - 4s 156ms/step - loss: 1.1706 - acc: 0.607
5 - val_loss: 1.3769 - val_acc: 0.5407
Accuracy:  0.5407319952774499
[0.6682408500590319, 0.6233766233766234, 0.7024793388429752, 0.51357733175915, 0.5
407319952774499]
```

In [16]:

```
t = 2.26 / np.sqrt(10)
e = (1-np.array(acc_scores1))-(1-np.array(acc_scores2))
stdtot =np.std(e)

dbar = np.mean(e)
print('model1 3X3 vs model1 2X2 acc range :',dbar-t*stdtot,dbar+t*stdtot)
```

```
model1 3X3 vs model1 2X2 acc range : -0.14373989683716187 -0.08010898155717108
```

Becasue the range is not include 0, so we can say that with 95% confident level, model1 3X3 and model1 2X2 are statisitcally different base on accuracy.

In [17]:

```
from statistics import mean
print('Average accuracy for model1 3X3 ',mean(acc_scores1))
print('Average accuracy for model1 2X2 ',mean(acc_scores2))
```

```
Average accuracy for model1 3X3  0.7216056670602126
Average accuracy for model1 2X2  0.609681227863046
```

Base on my statistics comparision, there is different between model1 3X3 and model1 2X2, and 3X3 have higher average accuracy score. So model1 3X3 is better.

Second model 3X3 kernal

In [69]:

```
#second model, this one is sample cnn
def create_cnn3():
    cnn3 = Sequential()
    # let's start with an AlexNet style convolutional phase
    cnn3.add(Conv2D(filters=32,
                    input_shape = (img_wh,img_wh,1),
                    kernel_size=(3,3),
                    padding='same',
                    activation='relu', data_format="channels_last")) # more compact syntax

    # no max pool before next conv layer!!
    cnn3.add(Conv2D(filters=64,
                    kernel_size=(3,3),
                    padding='same',
                    activation='relu')) # more compact syntax
    cnn3.add(MaxPooling2D(pool_size=(2, 2), data_format="channels_last"))


    # add one layer on flattened output
    cnn3.add(Dropout(0.25)) # add some dropout for regularization after conv layers
    cnn3.add(Flatten())
    cnn3.add(Dense(128, activation='relu'))
    cnn3.add(Dropout(0.5)) # add some dropout for regularization, again!
    cnn3.add(Dense(NUM_CLASSES, activation='softmax'))
    # Let's train the model
    cnn3.compile(loss='categorical_crossentropy', # 'categorical_crossentropy' 'mean_squared_err
or'
                optimizer='rmsprop', # 'adadelta' 'rmsprop'
                metrics=['accuracy'])
    return cnn3

cnn3 = create_cnn3()
```

In [70]:

```
# the flow method yields batches of images indefinitely, with the given transofmrations
history3 = cnn3.fit_generator(datagen.flow(X_train, y_train_ohe, batch_size=128),
                 steps_per_epoch=int(len(X_train)/128), # how many generators to go through per
epoch
                 epochs=50, verbose=1,
                 validation_data=(X_test,y_test_ohe),
                 callbacks=[EarlyStopping(monitor='val_loss', patience=4)]
                )
```

```
Epoch 1/50
26/26 [==============================] - 5s 198ms/step - loss: 3.5954 - acc: 0.137
3 - val_loss: 2.1242 - val_acc: 0.3282
Epoch 2/50
26/26 [==============================] - 4s 141ms/step - loss: 2.0795 - acc: 0.240
8 - val_loss: 1.9140 - val_acc: 0.3766
Epoch 3/50
26/26 [==============================] - 4s 141ms/step - loss: 1.9288 - acc: 0.292
7 - val_loss: 1.6872 - val_acc: 0.3979
Epoch 4/50
26/26 [==============================] - 4s 141ms/step - loss: 1.8412 - acc: 0.329
2 - val_loss: 1.4478 - val_acc: 0.4994
Epoch 5/50
26/26 [==============================] - 4s 141ms/step - loss: 1.7353 - acc: 0.374
3 - val_loss: 1.4864 - val_acc: 0.4604
Epoch 6/50
26/26 [==============================] - 4s 141ms/step - loss: 1.7105 - acc: 0.384
0 - val_loss: 1.4521 - val_acc: 0.5065
Epoch 7/50
26/26 [==============================] - 4s 141ms/step - loss: 1.5585 - acc: 0.426
6 - val_loss: 1.1685 - val_acc: 0.5773
Epoch 8/50
26/26 [==============================] - 4s 141ms/step - loss: 1.4349 - acc: 0.461
6 - val_loss: 1.0432 - val_acc: 0.6116
Epoch 9/50
26/26 [==============================] - 4s 141ms/step - loss: 1.4260 - acc: 0.465
2 - val_loss: 0.9659 - val_acc: 0.6446
Epoch 10/50
26/26 [==============================] - 4s 141ms/step - loss: 1.3650 - acc: 0.508
4 - val_loss: 1.1365 - val_acc: 0.5762
Epoch 11/50
26/26 [==============================] - 4s 142ms/step - loss: 1.2802 - acc: 0.521
7 - val_loss: 0.9245 - val_acc: 0.6600
Epoch 12/50
26/26 [==============================] - 4s 141ms/step - loss: 1.3076 - acc: 0.519
3 - val_loss: 1.0723 - val_acc: 0.6021
Epoch 13/50
26/26 [==============================] - 4s 141ms/step - loss: 1.2355 - acc: 0.541
3 - val_loss: 0.9844 - val_acc: 0.6246
Epoch 14/50
26/26 [==============================] - 4s 142ms/step - loss: 1.1887 - acc: 0.549
0 - val_loss: 0.8775 - val_acc: 0.6399
Epoch 15/50
26/26 [==============================] - 4s 141ms/step - loss: 1.1735 - acc: 0.562
3 - val_loss: 0.8377 - val_acc: 0.6612
Epoch 16/50
26/26 [==============================] - 4s 142ms/step - loss: 1.1154 - acc: 0.577
1 - val_loss: 0.9705 - val_acc: 0.6246
Epoch 17/50
26/26 [==============================] - 4s 142ms/step - loss: 1.0911 - acc: 0.584
6 - val_loss: 0.7299 - val_acc: 0.7084
Epoch 18/50
26/26 [==============================] - 4s 141ms/step - loss: 1.0914 - acc: 0.579
8 - val_loss: 0.8404 - val_acc: 0.6494
Epoch 19/50
26/26 [==============================] - 4s 142ms/step - loss: 1.0272 - acc: 0.598
7 - val_loss: 0.6851 - val_acc: 0.7084
Epoch 20/50
26/26 [==============================] - 4s 142ms/step - loss: 0.9987 - acc: 0.613
1 - val_loss: 1.2813 - val_acc: 0.5159
Epoch 21/50
```

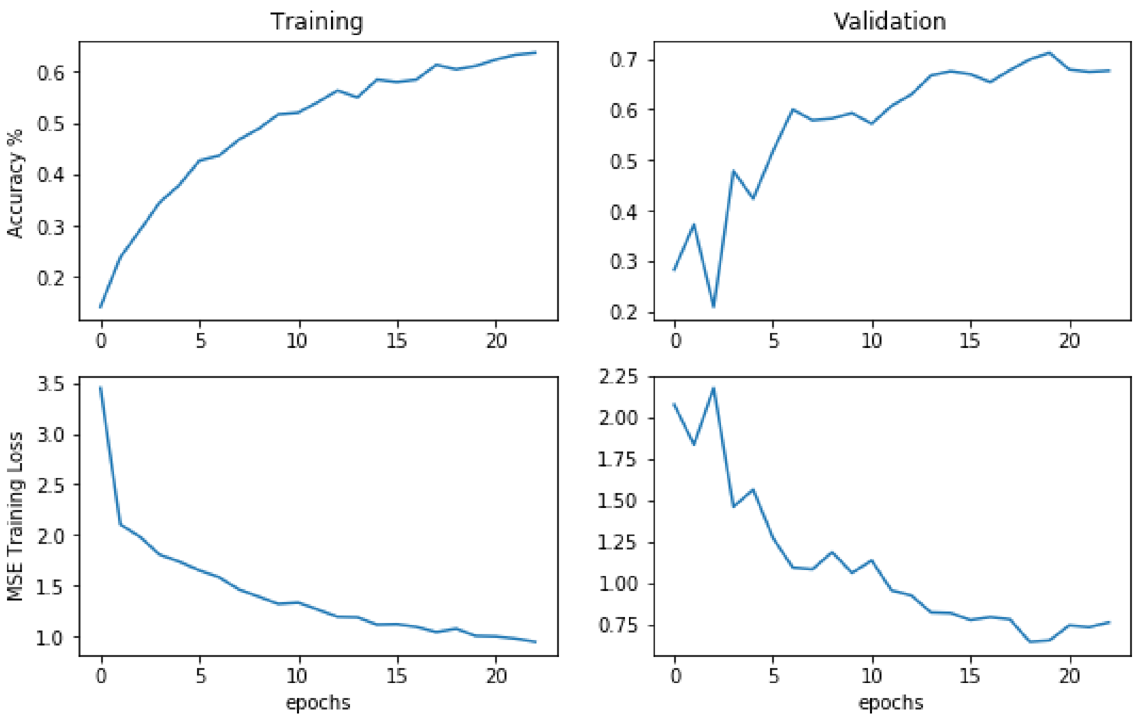
```
26/26 [==============================] - 4s 142ms/step - loss: 1.0568 - acc: 0.601
2 - val_loss: 0.6962 - val_acc: 0.7037
Epoch 22/50
26/26 [==============================] - 4s 143ms/step - loss: 0.9449 - acc: 0.634
5 - val_loss: 0.6509 - val_acc: 0.7344
Epoch 23/50
26/26 [==============================] - 4s 144ms/step - loss: 0.9454 - acc: 0.627
5 - val_loss: 0.6798 - val_acc: 0.7119
Epoch 24/50
26/26 [==============================] - 4s 143ms/step - loss: 0.9490 - acc: 0.633
6 - val_loss: 0.6302 - val_acc: 0.7344
Epoch 25/50
26/26 [==============================] - 4s 142ms/step - loss: 0.9196 - acc: 0.636
3 - val_loss: 0.5844 - val_acc: 0.7391
Epoch 26/50
26/26 [==============================] - 4s 144ms/step - loss: 0.9399 - acc: 0.633
8 - val_loss: 0.7153 - val_acc: 0.6989
Epoch 27/50
26/26 [==============================] - 4s 144ms/step - loss: 0.9011 - acc: 0.645
0 - val_loss: 0.6502 - val_acc: 0.7166
Epoch 28/50
26/26 [==============================] - 4s 146ms/step - loss: 0.9021 - acc: 0.643
6 - val_loss: 0.5895 - val_acc: 0.7332
Epoch 29/50
26/26 [==============================] - 4s 143ms/step - loss: 0.8656 - acc: 0.652
8 - val_loss: 0.5621 - val_acc: 0.7568
Epoch 30/50
26/26 [==============================] - 4s 143ms/step - loss: 0.8842 - acc: 0.656
8 - val_loss: 0.6031 - val_acc: 0.7237
Epoch 31/50
26/26 [==============================] - 4s 147ms/step - loss: 0.8569 - acc: 0.669
7 - val_loss: 0.5738 - val_acc: 0.7521
Epoch 32/50
26/26 [==============================] - 4s 143ms/step - loss: 0.8145 - acc: 0.673
3 - val_loss: 0.6388 - val_acc: 0.7249
Epoch 33/50
26/26 [==============================] - 4s 143ms/step - loss: 0.8298 - acc: 0.670
6 - val_loss: 0.6237 - val_acc: 0.7285
```

In [16]:

```
from matplotlib import pyplot as plt

%matplotlib inline

plt.figure(figsize=(10,6))
plt.subplot(2,2,1)
plt.plot(history3.history['acc'])

plt.ylabel('Accuracy %')
plt.title('Training')
plt.subplot(2,2,2)
plt.plot(history3.history['val_acc'])
plt.title('Validation')

plt.subplot(2,2,3)
plt.plot(history3.history['loss'])
plt.ylabel('MSE Training Loss')
plt.xlabel('epochs')

plt.subplot(2,2,4)
plt.plot(history3.history['val_loss'])
plt.xlabel('epochs')
```

Out[16]:

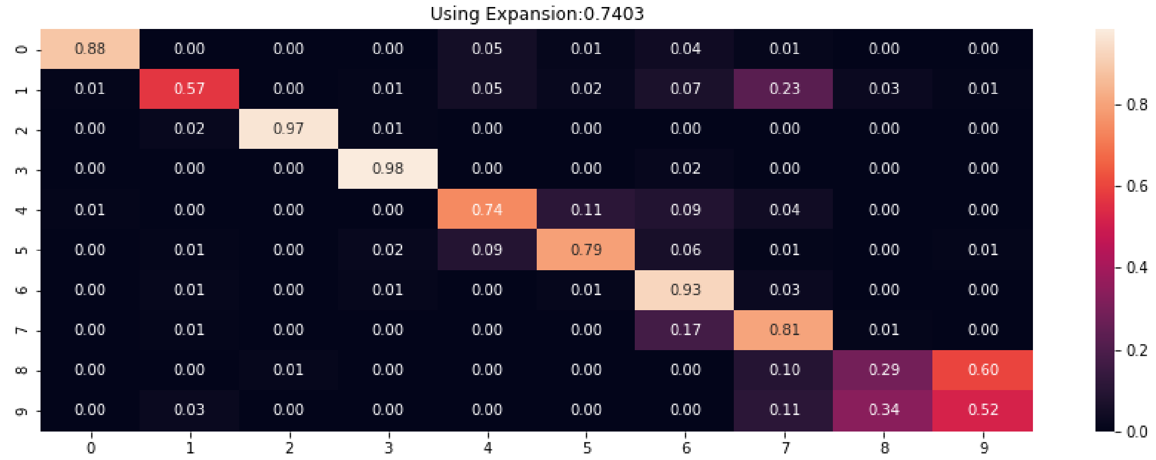
Text(0.5, 0, 'epochs')



For train vs loss, I think it converge, the line reach the bottom. For validation, the line is boucing at 18 epochs. But in general, they converge.

In [43]:

```
#    0- Apple_Black_rot',
#    1- Apple_healthy',
#    2- Grape_Black_rot',
#    3- Grape_healthy',
#    4- Pepper_Bacterial_spot',
#    5- Pepper_healthy',
#    6- Potato_healthy',
#    7- Potato_Late_blight',
#    8- Tomato_Bacterial_spot',
#    9- Tomato_healthy']

summarize_net(cnn3, X_test, y_test, title_text='Using Expansion:')
```



In [13]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#
sklearn.model_selection.StratifiedKFold.split
#https://github.com/Thakugan/machine-learning-notebooks/blob/master/6-wide-and-deep-networks/mus
hroom-hunting.ipynb
#I used this in last lab, just modified to new version
from sklearn.model_selection import StratifiedKFold
num_folds = 5

acc_scores3 =[]
skf3 = StratifiedKFold(n_splits=num_folds, shuffle=True)
for i, (train, test) in enumerate(skf3.split(X, y)):#here I used corss validation on my model

    cnn3 = create_cnn3()
    #doing modeling same as above, without for loop

    cnn3.fit_generator(datagen.flow(X_train, y_train_ohe, batch_size=128),
                      steps_per_epoch=int(len(X_train)/128), # how many generators to go through per
epoch
                      epochs=50, verbose=1,
                      validation_data=(X_test,y_test_ohe),
                      callbacks=[EarlyStopping(monitor='val_loss', patience=4)]
                     )
#this is just what i do without cross validation
    yhat = np.argmax(cnn3.predict(X_test), axis=1)

    acc_score3 = mt.accuracy_score(y_test, yhat)
    acc_scores3.append(acc_score3)#append all acc for k-fold
    print("Accuracy: ", acc_score3)
print(acc_scores3)
```

```
WARNING:tensorflow:From D:\APP\conda\lib\site-packages\tensorflow\python\ops\math_
ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will
be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/50
26/26 [==============================] - 8s 324ms/step - loss: 3.4801 - acc: 0.147
5 - val_loss: 2.0691 - val_acc: 0.2645
Epoch 2/50
26/26 [==============================] - 5s 179ms/step - loss: 2.0499 - acc: 0.245
6 - val_loss: 2.2862 - val_acc: 0.1913
Epoch 3/50
26/26 [==============================] - 4s 152ms/step - loss: 1.9626 - acc: 0.283
0 - val_loss: 1.7589 - val_acc: 0.3695
Epoch 4/50
26/26 [==============================] - 4s 152ms/step - loss: 1.8184 - acc: 0.335
2 - val_loss: 1.6383 - val_acc: 0.4179
Epoch 5/50
26/26 [==============================] - 4s 152ms/step - loss: 1.7242 - acc: 0.368
5 - val_loss: 1.4987 - val_acc: 0.4427
Epoch 6/50
26/26 [==============================] - 4s 152ms/step - loss: 1.6195 - acc: 0.409
4 - val_loss: 1.4763 - val_acc: 0.4593
Epoch 7/50
26/26 [==============================] - 4s 152ms/step - loss: 1.5099 - acc: 0.450
2 - val_loss: 1.5017 - val_acc: 0.4475
Epoch 8/50
26/26 [==============================] - 4s 152ms/step - loss: 1.4404 - acc: 0.461
4 - val_loss: 1.1835 - val_acc: 0.5667
Epoch 9/50
26/26 [==============================] - 4s 153ms/step - loss: 1.3634 - acc: 0.502
3 - val_loss: 1.1059 - val_acc: 0.5809
Epoch 10/50
26/26 [==============================] - 4s 153ms/step - loss: 1.3058 - acc: 0.513
2 - val_loss: 1.1082 - val_acc: 0.5785
Epoch 11/50
26/26 [==============================] - 4s 153ms/step - loss: 1.2173 - acc: 0.549
9 - val_loss: 1.1146 - val_acc: 0.5821
Epoch 12/50
26/26 [==============================] - 4s 152ms/step - loss: 1.1734 - acc: 0.560
2 - val_loss: 1.9192 - val_acc: 0.4439
Epoch 13/50
26/26 [==============================] - 4s 152ms/step - loss: 1.1350 - acc: 0.588
2 - val_loss: 0.9076 - val_acc: 0.6387
Epoch 14/50
26/26 [==============================] - 4s 152ms/step - loss: 1.1192 - acc: 0.586
6 - val_loss: 0.7987 - val_acc: 0.6753
Epoch 15/50
26/26 [==============================] - 4s 153ms/step - loss: 1.0097 - acc: 0.611
5 - val_loss: 0.9738 - val_acc: 0.6068
Epoch 16/50
26/26 [==============================] - 4s 153ms/step - loss: 1.0553 - acc: 0.607
3 - val_loss: 1.0002 - val_acc: 0.6139
Epoch 17/50
26/26 [==============================] - 4s 153ms/step - loss: 0.9977 - acc: 0.621
8 - val_loss: 0.8209 - val_acc: 0.6694
Epoch 18/50
26/26 [==============================] - 4s 153ms/step - loss: 0.9457 - acc: 0.636
8 - val_loss: 0.7977 - val_acc: 0.6682
Epoch 19/50
26/26 [==============================] - 4s 153ms/step - loss: 0.9574 - acc: 0.635
```

```
1 - val_loss: 0.8026 - val_acc: 0.6789
Epoch 20/50
26/26 [==============================] - 4s 154ms/step - loss: 0.9519 - acc: 0.633
4 - val_loss: 0.8304 - val_acc: 0.6753
Epoch 21/50
26/26 [==============================] - 4s 153ms/step - loss: 0.8934 - acc: 0.648
8 - val_loss: 0.7079 - val_acc: 0.6989
Epoch 22/50
26/26 [==============================] - 4s 153ms/step - loss: 0.8895 - acc: 0.653
3 - val_loss: 0.7122 - val_acc: 0.7001
Epoch 23/50
26/26 [==============================] - 4s 153ms/step - loss: 0.8889 - acc: 0.649
6 - val_loss: 0.6759 - val_acc: 0.7084
Epoch 24/50
26/26 [==============================] - 4s 153ms/step - loss: 0.8645 - acc: 0.667
9 - val_loss: 0.6661 - val_acc: 0.7214
Epoch 25/50
26/26 [==============================] - 4s 153ms/step - loss: 0.8368 - acc: 0.667
8 - val_loss: 2.8894 - val_acc: 0.3353
Epoch 26/50
26/26 [==============================] - 4s 152ms/step - loss: 0.8844 - acc: 0.673
2 - val_loss: 0.6588 - val_acc: 0.7320
Epoch 27/50
26/26 [==============================] - 4s 156ms/step - loss: 0.7758 - acc: 0.689
0 - val_loss: 0.6643 - val_acc: 0.7214
Epoch 28/50
26/26 [==============================] - 4s 156ms/step - loss: 0.8246 - acc: 0.663
8 - val_loss: 0.6069 - val_acc: 0.7320
Epoch 29/50
26/26 [==============================] - 4s 154ms/step - loss: 0.7800 - acc: 0.682
7 - val_loss: 0.6455 - val_acc: 0.7320
Epoch 30/50
26/26 [==============================] - 4s 154ms/step - loss: 0.7970 - acc: 0.679
7 - val_loss: 0.6968 - val_acc: 0.6966
Epoch 31/50
26/26 [==============================] - 4s 152ms/step - loss: 0.7665 - acc: 0.685
3 - val_loss: 0.6154 - val_acc: 0.7367
Epoch 32/50
26/26 [==============================] - 4s 152ms/step - loss: 0.8230 - acc: 0.678
1 - val_loss: 0.6364 - val_acc: 0.7285
Accuracy:  0.7284533648170012
Epoch 1/50
26/26 [==============================] - 5s 196ms/step - loss: 3.2853 - acc: 0.124
7 - val_loss: 2.2734 - val_acc: 0.1263
Epoch 2/50
26/26 [==============================] - 4s 153ms/step - loss: 2.1405 - acc: 0.195
9 - val_loss: 2.0161 - val_acc: 0.2231
Epoch 3/50
26/26 [==============================] - 4s 153ms/step - loss: 2.0556 - acc: 0.252
2 - val_loss: 1.8656 - val_acc: 0.3483
Epoch 4/50
26/26 [==============================] - 4s 153ms/step - loss: 1.9003 - acc: 0.300
6 - val_loss: 1.6094 - val_acc: 0.4534
Epoch 5/50
26/26 [==============================] - 4s 153ms/step - loss: 1.8332 - acc: 0.339
3 - val_loss: 1.7026 - val_acc: 0.3636
Epoch 6/50
26/26 [==============================] - 4s 153ms/step - loss: 1.7518 - acc: 0.364
2 - val_loss: 1.4276 - val_acc: 0.4923
Epoch 7/50
26/26 [==============================] - 4s 152ms/step - loss: 1.6126 - acc: 0.415
```

```
1 - val_loss: 1.3069 - val_acc: 0.4994
Epoch 8/50
26/26 [==============================] - 4s 153ms/step - loss: 1.6374 - acc: 0.423
2 - val_loss: 1.4087 - val_acc: 0.4793
Epoch 9/50
26/26 [==============================] - 4s 153ms/step - loss: 1.5034 - acc: 0.457
6 - val_loss: 1.2806 - val_acc: 0.5136
Epoch 10/50
26/26 [==============================] - 4s 153ms/step - loss: 1.4725 - acc: 0.466
0 - val_loss: 1.2185 - val_acc: 0.5478
Epoch 11/50
26/26 [==============================] - 4s 153ms/step - loss: 1.3578 - acc: 0.502
4 - val_loss: 1.1304 - val_acc: 0.5537
Epoch 12/50
26/26 [==============================] - 4s 154ms/step - loss: 1.2937 - acc: 0.518
1 - val_loss: 1.1091 - val_acc: 0.5809
Epoch 13/50
26/26 [==============================] - 4s 152ms/step - loss: 1.2900 - acc: 0.530
7 - val_loss: 0.9068 - val_acc: 0.6375
Epoch 14/50
26/26 [==============================] - 4s 153ms/step - loss: 1.2098 - acc: 0.555
2 - val_loss: 0.9500 - val_acc: 0.6517
Epoch 15/50
26/26 [==============================] - 4s 153ms/step - loss: 1.2194 - acc: 0.551
5 - val_loss: 1.2601 - val_acc: 0.4888
Epoch 16/50
26/26 [==============================] - 4s 153ms/step - loss: 1.1464 - acc: 0.576
2 - val_loss: 1.2141 - val_acc: 0.5254
Epoch 17/50
26/26 [==============================] - 4s 153ms/step - loss: 1.1059 - acc: 0.597
0 - val_loss: 0.9327 - val_acc: 0.6446
Accuracy:  0.6446280991735537
Epoch 1/50
26/26 [==============================] - 5s 197ms/step - loss: 3.2447 - acc: 0.136
4 - val_loss: 2.1637 - val_acc: 0.3070
Epoch 2/50
26/26 [==============================] - 4s 152ms/step - loss: 2.1028 - acc: 0.226
6 - val_loss: 1.8880 - val_acc: 0.3412
Epoch 3/50
26/26 [==============================] - 4s 152ms/step - loss: 1.8924 - acc: 0.301
3 - val_loss: 2.6426 - val_acc: 0.1665
Epoch 4/50
26/26 [==============================] - 4s 152ms/step - loss: 1.8489 - acc: 0.328
1 - val_loss: 1.5983 - val_acc: 0.4274
Epoch 5/50
26/26 [==============================] - 4s 152ms/step - loss: 1.6998 - acc: 0.381
7 - val_loss: 1.7766 - val_acc: 0.3436
Epoch 6/50
26/26 [==============================] - 4s 152ms/step - loss: 1.6689 - acc: 0.411
5 - val_loss: 1.4864 - val_acc: 0.4758
Epoch 7/50
26/26 [==============================] - 4s 152ms/step - loss: 1.4927 - acc: 0.454
5 - val_loss: 1.3232 - val_acc: 0.5313
Epoch 8/50
26/26 [==============================] - 4s 152ms/step - loss: 1.3973 - acc: 0.481
5 - val_loss: 1.1625 - val_acc: 0.5691
Epoch 9/50
26/26 [==============================] - 4s 152ms/step - loss: 1.3619 - acc: 0.498
0 - val_loss: 1.2056 - val_acc: 0.5537
Epoch 10/50
26/26 [==============================] - 4s 153ms/step - loss: 1.3178 - acc: 0.511
```

```
5 - val_loss: 1.1958 - val_acc: 0.5431
Epoch 11/50
26/26 [==============================] - 4s 153ms/step - loss: 1.2616 - acc: 0.525
9 - val_loss: 1.0004 - val_acc: 0.6116
Epoch 12/50
26/26 [==============================] - 4s 153ms/step - loss: 1.1754 - acc: 0.566
9 - val_loss: 0.9983 - val_acc: 0.6293
Epoch 13/50
26/26 [==============================] - 4s 152ms/step - loss: 1.1817 - acc: 0.578
6 - val_loss: 0.9367 - val_acc: 0.6316
Epoch 14/50
26/26 [==============================] - 4s 152ms/step - loss: 1.0786 - acc: 0.590
2 - val_loss: 0.9365 - val_acc: 0.6198
Epoch 15/50
26/26 [==============================] - 4s 153ms/step - loss: 1.0644 - acc: 0.585
2 - val_loss: 0.9829 - val_acc: 0.6316
Epoch 16/50
26/26 [==============================] - 4s 152ms/step - loss: 1.0638 - acc: 0.598
9 - val_loss: 0.8392 - val_acc: 0.6682
Epoch 17/50
26/26 [==============================] - 4s 152ms/step - loss: 1.0299 - acc: 0.606
1 - val_loss: 0.8727 - val_acc: 0.6517
Epoch 18/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9969 - acc: 0.615
3 - val_loss: 0.7059 - val_acc: 0.7178
Epoch 19/50
26/26 [==============================] - 4s 153ms/step - loss: 0.9783 - acc: 0.628
5 - val_loss: 0.7537 - val_acc: 0.6942
Epoch 20/50
26/26 [==============================] - 4s 152ms/step - loss: 1.0083 - acc: 0.622
8 - val_loss: 0.7237 - val_acc: 0.7190
Epoch 21/50
26/26 [==============================] - 4s 153ms/step - loss: 0.9100 - acc: 0.648
1 - val_loss: 0.7335 - val_acc: 0.7155
Epoch 22/50
26/26 [==============================] - 4s 152ms/step - loss: 0.8672 - acc: 0.664
5 - val_loss: 0.7247 - val_acc: 0.7096
Accuracy:  0.7095631641086186
Epoch 1/50
26/26 [==============================] - 5s 199ms/step - loss: 3.5097 - acc: 0.130
1 - val_loss: 2.1853 - val_acc: 0.2479
Epoch 2/50
26/26 [==============================] - 4s 152ms/step - loss: 2.1219 - acc: 0.222
4 - val_loss: 1.9703 - val_acc: 0.2881
Epoch 3/50
26/26 [==============================] - 4s 153ms/step - loss: 1.9827 - acc: 0.274
1 - val_loss: 1.8197 - val_acc: 0.3518
Epoch 4/50
26/26 [==============================] - 4s 152ms/step - loss: 1.8941 - acc: 0.325
8 - val_loss: 1.5678 - val_acc: 0.4203
Epoch 5/50
26/26 [==============================] - 4s 152ms/step - loss: 1.7883 - acc: 0.358
6 - val_loss: 1.4613 - val_acc: 0.5065
Epoch 6/50
26/26 [==============================] - 4s 152ms/step - loss: 1.7370 - acc: 0.393
0 - val_loss: 1.5092 - val_acc: 0.4581
Epoch 7/50
26/26 [==============================] - 4s 152ms/step - loss: 1.5828 - acc: 0.428
9 - val_loss: 1.4444 - val_acc: 0.4911
Epoch 8/50
26/26 [==============================] - 4s 153ms/step - loss: 1.5458 - acc: 0.435
```

```
1 - val_loss: 1.1680 - val_acc: 0.6045
Epoch 9/50
26/26 [==============================] - 4s 153ms/step - loss: 1.4687 - acc: 0.474
8 - val_loss: 1.1436 - val_acc: 0.5962
Epoch 10/50
26/26 [==============================] - 4s 153ms/step - loss: 1.3786 - acc: 0.492
9 - val_loss: 1.0872 - val_acc: 0.6092
Epoch 11/50
26/26 [==============================] - 4s 152ms/step - loss: 1.2971 - acc: 0.525
4 - val_loss: 1.0491 - val_acc: 0.6305
Epoch 12/50
26/26 [==============================] - 4s 152ms/step - loss: 1.3085 - acc: 0.513
2 - val_loss: 1.1066 - val_acc: 0.5880
Epoch 13/50
26/26 [==============================] - 4s 152ms/step - loss: 1.2184 - acc: 0.563
7 - val_loss: 0.9632 - val_acc: 0.6116
Epoch 14/50
26/26 [==============================] - 4s 152ms/step - loss: 1.1960 - acc: 0.558
8 - val_loss: 0.8413 - val_acc: 0.6907
Epoch 15/50
26/26 [==============================] - 4s 152ms/step - loss: 1.1201 - acc: 0.591
4 - val_loss: 0.9549 - val_acc: 0.6458
Epoch 16/50
26/26 [==============================] - 4s 152ms/step - loss: 1.1268 - acc: 0.570
4 - val_loss: 0.9243 - val_acc: 0.6635
Epoch 17/50
26/26 [==============================] - 4s 153ms/step - loss: 1.0846 - acc: 0.587
9 - val_loss: 0.8461 - val_acc: 0.6836
Epoch 18/50
26/26 [==============================] - 4s 152ms/step - loss: 1.0666 - acc: 0.590
9 - val_loss: 0.7440 - val_acc: 0.7107
Epoch 19/50
26/26 [==============================] - 4s 152ms/step - loss: 1.0426 - acc: 0.598
8 - val_loss: 0.7805 - val_acc: 0.6848
Epoch 20/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9821 - acc: 0.613
7 - val_loss: 0.7916 - val_acc: 0.6966
Epoch 21/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9956 - acc: 0.615
5 - val_loss: 0.8127 - val_acc: 0.6706
Epoch 22/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9744 - acc: 0.627
8 - val_loss: 0.7363 - val_acc: 0.6942
Epoch 23/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9724 - acc: 0.626
9 - val_loss: 0.9119 - val_acc: 0.6399
Epoch 24/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9041 - acc: 0.648
8 - val_loss: 0.6584 - val_acc: 0.7344
Epoch 25/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9521 - acc: 0.629
5 - val_loss: 0.6529 - val_acc: 0.7355
Epoch 26/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9029 - acc: 0.642
4 - val_loss: 0.7697 - val_acc: 0.6812
Epoch 27/50
26/26 [==============================] - 4s 153ms/step - loss: 0.8694 - acc: 0.659
2 - val_loss: 2.2219 - val_acc: 0.4451
Epoch 28/50
26/26 [==============================] - 4s 155ms/step - loss: 0.9863 - acc: 0.642
4 - val_loss: 0.6022 - val_acc: 0.7473
```

```
Epoch 29/50
26/26 [==============================] - 4s 152ms/step - loss: 0.8308 - acc: 0.669
2 - val_loss: 0.6626 - val_acc: 0.7202
Epoch 30/50
26/26 [==============================] - 4s 153ms/step - loss: 0.8447 - acc: 0.667
7 - val_loss: 0.8568 - val_acc: 0.6694
Epoch 31/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9037 - acc: 0.648
9 - val_loss: 0.6133 - val_acc: 0.7344
Epoch 32/50
26/26 [==============================] - 4s 153ms/step - loss: 0.7718 - acc: 0.680
5 - val_loss: 0.5525 - val_acc: 0.7568
Epoch 33/50
26/26 [==============================] - 4s 153ms/step - loss: 0.8213 - acc: 0.670
9 - val_loss: 0.7635 - val_acc: 0.6824
Epoch 34/50
26/26 [==============================] - 4s 153ms/step - loss: 0.8025 - acc: 0.690
1 - val_loss: 0.7157 - val_acc: 0.7013
Epoch 35/50
26/26 [==============================] - 4s 152ms/step - loss: 0.8165 - acc: 0.678
3 - val_loss: 0.6802 - val_acc: 0.7119
Epoch 36/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9039 - acc: 0.646
9 - val_loss: 0.5856 - val_acc: 0.7485
Accuracy:  0.7485242030696576
Epoch 1/50
26/26 [==============================] - 5s 200ms/step - loss: 3.1917 - acc: 0.137
6 - val_loss: 2.0615 - val_acc: 0.2621
Epoch 2/50
26/26 [==============================] - 4s 153ms/step - loss: 2.1280 - acc: 0.211
3 - val_loss: 2.0253 - val_acc: 0.1983
Epoch 3/50
26/26 [==============================] - 4s 152ms/step - loss: 2.0153 - acc: 0.266
1 - val_loss: 1.8254 - val_acc: 0.2999
Epoch 4/50
26/26 [==============================] - 4s 152ms/step - loss: 1.9212 - acc: 0.306
2 - val_loss: 1.6599 - val_acc: 0.3908
Epoch 5/50
26/26 [==============================] - 4s 153ms/step - loss: 1.7585 - acc: 0.379
6 - val_loss: 1.9097 - val_acc: 0.3259
Epoch 6/50
26/26 [==============================] - 4s 152ms/step - loss: 1.7360 - acc: 0.375
2 - val_loss: 1.4293 - val_acc: 0.5006
Epoch 7/50
26/26 [==============================] - 4s 152ms/step - loss: 1.5749 - acc: 0.434
7 - val_loss: 1.4259 - val_acc: 0.4817
Epoch 8/50
26/26 [==============================] - 4s 152ms/step - loss: 1.5331 - acc: 0.444
2 - val_loss: 1.4447 - val_acc: 0.4746
Epoch 9/50
26/26 [==============================] - 4s 153ms/step - loss: 1.4553 - acc: 0.466
1 - val_loss: 1.1745 - val_acc: 0.5832
Epoch 10/50
26/26 [==============================] - 4s 152ms/step - loss: 1.3553 - acc: 0.497
5 - val_loss: 1.1014 - val_acc: 0.6021
Epoch 11/50
26/26 [==============================] - 4s 152ms/step - loss: 1.3553 - acc: 0.508
2 - val_loss: 1.0624 - val_acc: 0.6045
Epoch 12/50
26/26 [==============================] - 4s 153ms/step - loss: 1.3251 - acc: 0.509
6 - val_loss: 1.3402 - val_acc: 0.4935
```

```
Epoch 13/50
26/26 [==============================] - 4s 153ms/step - loss: 1.2763 - acc: 0.521
3 - val_loss: 1.1424 - val_acc: 0.5903
Epoch 14/50
26/26 [==============================] - 4s 153ms/step - loss: 1.2198 - acc: 0.543
7 - val_loss: 1.2121 - val_acc: 0.5762
Epoch 15/50
26/26 [==============================] - 4s 153ms/step - loss: 1.1659 - acc: 0.558
2 - val_loss: 1.0557 - val_acc: 0.6068
Epoch 16/50
26/26 [==============================] - 4s 153ms/step - loss: 1.1896 - acc: 0.555
5 - val_loss: 0.9608 - val_acc: 0.6564
Epoch 17/50
26/26 [==============================] - 4s 152ms/step - loss: 1.0865 - acc: 0.594
4 - val_loss: 0.8165 - val_acc: 0.6895
Epoch 18/50
26/26 [==============================] - 4s 152ms/step - loss: 1.1237 - acc: 0.577
9 - val_loss: 0.9231 - val_acc: 0.6375
Epoch 19/50
26/26 [==============================] - 4s 153ms/step - loss: 1.0446 - acc: 0.610
0 - val_loss: 0.8468 - val_acc: 0.6635
Epoch 20/50
26/26 [==============================] - 4s 153ms/step - loss: 1.0782 - acc: 0.591
1 - val_loss: 0.8998 - val_acc: 0.6352
Epoch 21/50
26/26 [==============================] - 4s 152ms/step - loss: 1.0378 - acc: 0.600
3 - val_loss: 0.7834 - val_acc: 0.6942
Epoch 22/50
26/26 [==============================] - 4s 153ms/step - loss: 1.0313 - acc: 0.594
6 - val_loss: 0.7404 - val_acc: 0.7096
Epoch 23/50
26/26 [==============================] - 4s 153ms/step - loss: 0.9883 - acc: 0.612
7 - val_loss: 0.7631 - val_acc: 0.6800
Epoch 24/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9866 - acc: 0.629
8 - val_loss: 0.8881 - val_acc: 0.6576
Epoch 25/50
26/26 [==============================] - 4s 153ms/step - loss: 1.0153 - acc: 0.613
6 - val_loss: 0.8958 - val_acc: 0.6600
Epoch 26/50
26/26 [==============================] - 4s 152ms/step - loss: 0.9754 - acc: 0.631
6 - val_loss: 0.8746 - val_acc: 0.6635
Accuracy:  0.6635182998819362
[0.7284533648170012, 0.6446280991735537, 0.7095631641086186, 0.7485242030696576,
0.6635182998819362]
```

Second model 2X2 kernal

In [11]:

```
#second model, this one is sample cnn
def create_cnn4():
    cnn4 = Sequential()
    # let's start with an AlexNet style convolutional phase
    cnn4.add(Conv2D(filters=32,
                    input_shape = (img_wh,img_wh,1),
                    kernel_size=(2,2),
                    padding='same',
                    activation='relu', data_format="channels_last")) # more compact syntax

    # no max pool before next conv layer!!
    cnn4.add(Conv2D(filters=64,
                    kernel_size=(2,2),
                    padding='same',
                    activation='relu')) # more compact syntax
    cnn4.add(MaxPooling2D(pool_size=(2, 2), data_format="channels_last"))


    # add one layer on flattened output
    cnn4.add(Dropout(0.25)) # add some dropout for regularization after conv layers
    cnn4.add(Flatten())
    cnn4.add(Dense(128, activation='relu'))
    cnn4.add(Dropout(0.5)) # add some dropout for regularization, again!
    cnn4.add(Dense(NUM_CLASSES, activation='softmax'))
    # Let's train the model
    cnn4.compile(loss='categorical_crossentropy', # 'categorical_crossentropy' 'mean_squared_err
or'
                  optimizer='rmsprop', # 'adadelta' 'rmsprop'
                  metrics=['accuracy'])
    return cnn4

cnn4 = create_cnn4()
```

```
WARNING:tensorflow:From D:\APP\conda\lib\site-packages\tensorflow\python\framework
\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is de
precated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\APP\conda\lib\site-packages\keras\backend\tensorflow_ba
ckend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob
is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_p
rob`.
```

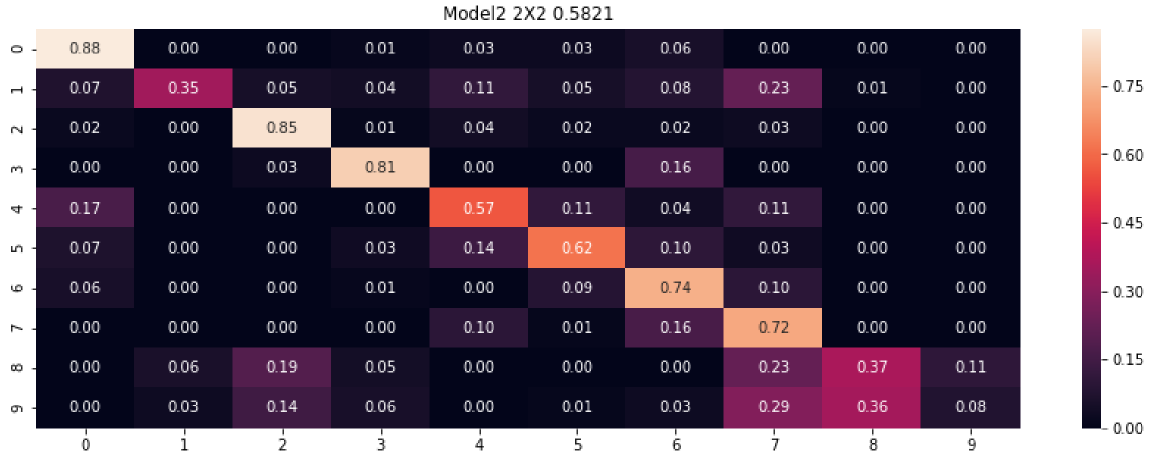
In [14]:

```
# the flow method yields batches of images indefinitely, with the given transofmrations
history4 = cnn4.fit_generator(datagen.flow(X_train, y_train_ohe, batch_size=128),
                   steps_per_epoch=int(len(X_train)/128), # how many generators to go through per
epoch
                   epochs=50, verbose=1,
                   validation_data=(X_test,y_test_ohe),
                   callbacks=[EarlyStopping(monitor='val_loss', patience=4)]
                  )
```

```
WARNING:tensorflow:From D:\APP\conda\lib\site-packages\tensorflow\python\ops\math_
ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will
be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/50
26/26 [==============================] - 7s 276ms/step - loss: 4.1019 - acc: 0.146
0 - val_loss: 1.9789 - val_acc: 0.3459
Epoch 2/50
26/26 [==============================] - 4s 143ms/step - loss: 2.0559 - acc: 0.260
6 - val_loss: 1.8714 - val_acc: 0.3483
Epoch 3/50
26/26 [==============================] - 3s 128ms/step - loss: 1.9218 - acc: 0.310
0 - val_loss: 2.2367 - val_acc: 0.1464
Epoch 4/50
26/26 [==============================] - 3s 129ms/step - loss: 1.7808 - acc: 0.354
7 - val_loss: 1.6589 - val_acc: 0.4109
Epoch 5/50
26/26 [==============================] - 3s 127ms/step - loss: 1.6930 - acc: 0.385
8 - val_loss: 1.4546 - val_acc: 0.4522
Epoch 6/50
26/26 [==============================] - 3s 126ms/step - loss: 1.6558 - acc: 0.406
9 - val_loss: 1.4562 - val_acc: 0.4569
Epoch 7/50
26/26 [==============================] - 3s 126ms/step - loss: 1.6009 - acc: 0.415
3 - val_loss: 1.1471 - val_acc: 0.5785
Epoch 8/50
26/26 [==============================] - 3s 126ms/step - loss: 1.5435 - acc: 0.448
6 - val_loss: 1.1649 - val_acc: 0.5738
Epoch 9/50
26/26 [==============================] - 3s 126ms/step - loss: 1.4839 - acc: 0.454
9 - val_loss: 1.2147 - val_acc: 0.5514
Epoch 10/50
26/26 [==============================] - 3s 126ms/step - loss: 1.4479 - acc: 0.470
2 - val_loss: 1.3285 - val_acc: 0.5041
Epoch 11/50
26/26 [==============================] - 3s 126ms/step - loss: 1.4565 - acc: 0.471
3 - val_loss: 1.1667 - val_acc: 0.5821
```

In [23]:

```
#   0- Apple_Black_rot',
#   1- Apple_healthy',
#   2- Grape_Black_rot',
#   3- Grape_healthy',
#   4- Pepper_Bacterial_spot',
#   5- Pepper_healthy',
#   6- Potato_healthy',
#   7- Potato_Late_blight',
#   8- Tomato_Bacterial_spot',
#   9- Tomato_healthy']

summarize_net(cnn4, X_test, y_test, title_text='Model2 2X2 ')
```



In [12]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#
sklearn.model_selection.StratifiedKFold.split
#https://github.com/Thakugan/machine-learning-notebooks/blob/master/6-wide-and-deep-networks/mus
hroom-hunting.ipynb
#I used this in last lab, just modified to new version
from sklearn.model_selection import StratifiedKFold
num_folds = 5

acc_scores4 =[]
skf4 = StratifiedKFold(n_splits=num_folds, shuffle=True)
for i, (train, test) in enumerate(skf4.split(X, y)):#here I used corss validation on my model

    cnn4 = create_cnn4()
    #doing modeling same as above, without for loop

    cnn4.fit_generator(datagen.flow(X_train, y_train_ohe, batch_size=128),
                  steps_per_epoch=int(len(X_train)/128), # how many generators to go through per
epoch
                  epochs=50, verbose=1,
                  validation_data=(X_test,y_test_ohe),
                  callbacks=[EarlyStopping(monitor='val_loss', patience=4)]
                 )
#this is just what i do without cross validation
    yhat = np.argmax(cnn4.predict(X_test), axis=1)

    acc_score4 = mt.accuracy_score(y_test, yhat)
    acc_scores4.append(acc_score4)#append all acc for k-fold
    print("Accuracy: ", acc_score4)
print(acc_scores4)
```

```
WARNING:tensorflow:From D:\APP\conda\lib\site-packages\tensorflow\python\ops\math_
ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will
be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/50
26/26 [==============================] - 8s 314ms/step - loss: 3.6436 - acc: 0.116
3 - val_loss: 2.2099 - val_acc: 0.1169
Epoch 2/50
26/26 [==============================] - 4s 143ms/step - loss: 2.1777 - acc: 0.194
8 - val_loss: 1.9925 - val_acc: 0.2409
Epoch 3/50
26/26 [==============================] - 3s 125ms/step - loss: 2.1202 - acc: 0.224
4 - val_loss: 1.9829 - val_acc: 0.1960
Epoch 4/50
26/26 [==============================] - 3s 126ms/step - loss: 1.9789 - acc: 0.277
0 - val_loss: 1.7650 - val_acc: 0.3861
Epoch 5/50
26/26 [==============================] - 3s 126ms/step - loss: 1.9023 - acc: 0.296
9 - val_loss: 1.6447 - val_acc: 0.4368
Epoch 6/50
26/26 [==============================] - 3s 126ms/step - loss: 1.8110 - acc: 0.335
6 - val_loss: 2.0216 - val_acc: 0.2727
Epoch 7/50
26/26 [==============================] - 3s 126ms/step - loss: 1.7766 - acc: 0.361
2 - val_loss: 1.5024 - val_acc: 0.4262
Epoch 8/50
26/26 [==============================] - 3s 125ms/step - loss: 1.7752 - acc: 0.350
0 - val_loss: 1.3985 - val_acc: 0.5360
Epoch 9/50
26/26 [==============================] - 3s 126ms/step - loss: 1.6708 - acc: 0.383
7 - val_loss: 1.5919 - val_acc: 0.4168
Epoch 10/50
26/26 [==============================] - 3s 126ms/step - loss: 1.6162 - acc: 0.405
8 - val_loss: 1.2334 - val_acc: 0.5561
Epoch 11/50
26/26 [==============================] - 3s 126ms/step - loss: 1.5697 - acc: 0.426
1 - val_loss: 1.1602 - val_acc: 0.5608
Epoch 12/50
26/26 [==============================] - 3s 126ms/step - loss: 1.5684 - acc: 0.427
5 - val_loss: 1.2195 - val_acc: 0.5738
Epoch 13/50
26/26 [==============================] - 3s 126ms/step - loss: 1.4882 - acc: 0.445
7 - val_loss: 1.5846 - val_acc: 0.4191
Epoch 14/50
26/26 [==============================] - 3s 125ms/step - loss: 1.4590 - acc: 0.457
2 - val_loss: 1.9573 - val_acc: 0.3601
Epoch 15/50
26/26 [==============================] - 3s 126ms/step - loss: 1.4594 - acc: 0.471
1 - val_loss: 1.0495 - val_acc: 0.5868
Epoch 16/50
26/26 [==============================] - 3s 125ms/step - loss: 1.4007 - acc: 0.482
4 - val_loss: 1.1174 - val_acc: 0.5431
Epoch 17/50
26/26 [==============================] - 3s 126ms/step - loss: 1.4158 - acc: 0.483
3 - val_loss: 1.0664 - val_acc: 0.5832
Epoch 18/50
26/26 [==============================] - 3s 126ms/step - loss: 1.3417 - acc: 0.499
6 - val_loss: 1.1898 - val_acc: 0.5502
Epoch 19/50
26/26 [==============================] - 3s 126ms/step - loss: 1.3609 - acc: 0.494
```

```
7 - val_loss: 1.0826 - val_acc: 0.5679
Accuracy:  0.5678866587957497
Epoch 1/50
26/26 [==============================] - 4s 168ms/step - loss: 3.6339 - acc: 0.134
0 - val_loss: 2.1380 - val_acc: 0.2031
Epoch 2/50
26/26 [==============================] - 3s 126ms/step - loss: 2.1296 - acc: 0.199
4 - val_loss: 2.0394 - val_acc: 0.2597
Epoch 3/50
26/26 [==============================] - 3s 125ms/step - loss: 2.0582 - acc: 0.237
6 - val_loss: 1.7982 - val_acc: 0.3081
Epoch 4/50
26/26 [==============================] - 3s 126ms/step - loss: 1.9969 - acc: 0.262
1 - val_loss: 1.7705 - val_acc: 0.4026
Epoch 5/50
26/26 [==============================] - 3s 126ms/step - loss: 1.8733 - acc: 0.322
1 - val_loss: 1.7225 - val_acc: 0.3684
Epoch 6/50
26/26 [==============================] - 3s 126ms/step - loss: 1.8363 - acc: 0.327
4 - val_loss: 1.5146 - val_acc: 0.4758
Epoch 7/50
26/26 [==============================] - 3s 126ms/step - loss: 1.7654 - acc: 0.341
6 - val_loss: 1.5928 - val_acc: 0.4026
Epoch 8/50
26/26 [==============================] - 3s 126ms/step - loss: 1.7091 - acc: 0.378
1 - val_loss: 1.4066 - val_acc: 0.4734
Epoch 9/50
26/26 [==============================] - 3s 126ms/step - loss: 1.6750 - acc: 0.377
9 - val_loss: 1.3361 - val_acc: 0.5053
Epoch 10/50
26/26 [==============================] - 3s 126ms/step - loss: 1.6208 - acc: 0.408
5 - val_loss: 1.4563 - val_acc: 0.4368
Epoch 11/50
26/26 [==============================] - 3s 126ms/step - loss: 1.5742 - acc: 0.417
8 - val_loss: 1.2082 - val_acc: 0.5620
Epoch 12/50
26/26 [==============================] - 3s 127ms/step - loss: 1.4946 - acc: 0.435
6 - val_loss: 1.6897 - val_acc: 0.3896
Epoch 13/50
26/26 [==============================] - 4s 138ms/step - loss: 1.5367 - acc: 0.434
4 - val_loss: 1.0966 - val_acc: 0.5962
Epoch 14/50
26/26 [==============================] - 3s 133ms/step - loss: 1.4625 - acc: 0.456
5 - val_loss: 1.0483 - val_acc: 0.6139
Epoch 15/50
26/26 [==============================] - 3s 128ms/step - loss: 1.4213 - acc: 0.465
8 - val_loss: 1.0547 - val_acc: 0.5832
Epoch 16/50
26/26 [==============================] - 3s 128ms/step - loss: 1.3922 - acc: 0.491
1 - val_loss: 1.5475 - val_acc: 0.4746
Epoch 17/50
26/26 [==============================] - 3s 129ms/step - loss: 1.4017 - acc: 0.472
3 - val_loss: 1.0766 - val_acc: 0.5738
Epoch 18/50
26/26 [==============================] - 3s 127ms/step - loss: 1.3278 - acc: 0.502
8 - val_loss: 1.1845 - val_acc: 0.5396
Accuracy:  0.5395513577331759
Epoch 1/50
26/26 [==============================] - 4s 169ms/step - loss: 4.0963 - acc: 0.123
5 - val_loss: 2.1809 - val_acc: 0.2609
Epoch 2/50
```

```
26/26 [==============================] - 3s 128ms/step - loss: 2.1989 - acc: 0.183
0 - val_loss: 1.9632 - val_acc: 0.2349
Epoch 3/50
26/26 [==============================] - 3s 129ms/step - loss: 2.0582 - acc: 0.239
9 - val_loss: 1.8373 - val_acc: 0.3778
Epoch 4/50
26/26 [==============================] - 3s 127ms/step - loss: 1.9489 - acc: 0.278
4 - val_loss: 1.8727 - val_acc: 0.3011
Epoch 5/50
26/26 [==============================] - 3s 126ms/step - loss: 1.8823 - acc: 0.312
7 - val_loss: 1.5782 - val_acc: 0.4262
Epoch 6/50
26/26 [==============================] - 3s 127ms/step - loss: 1.7961 - acc: 0.347
8 - val_loss: 1.4892 - val_acc: 0.4274
Epoch 7/50
26/26 [==============================] - 3s 126ms/step - loss: 1.7677 - acc: 0.370
0 - val_loss: 1.5869 - val_acc: 0.4404
Epoch 8/50
26/26 [==============================] - 3s 127ms/step - loss: 1.6681 - acc: 0.398
7 - val_loss: 1.3036 - val_acc: 0.5490
Epoch 9/50
26/26 [==============================] - 3s 130ms/step - loss: 1.5994 - acc: 0.401
4 - val_loss: 1.2465 - val_acc: 0.5360
Epoch 10/50
26/26 [==============================] - 3s 126ms/step - loss: 1.5798 - acc: 0.423
4 - val_loss: 1.2662 - val_acc: 0.5183
Epoch 11/50
26/26 [==============================] - 3s 126ms/step - loss: 1.5389 - acc: 0.441
2 - val_loss: 1.2362 - val_acc: 0.5549
Epoch 12/50
26/26 [==============================] - 3s 127ms/step - loss: 1.4922 - acc: 0.451
1 - val_loss: 1.3006 - val_acc: 0.5207
Epoch 13/50
26/26 [==============================] - 3s 129ms/step - loss: 1.4809 - acc: 0.450
0 - val_loss: 1.0606 - val_acc: 0.5915
Epoch 14/50
26/26 [==============================] - 3s 128ms/step - loss: 1.4203 - acc: 0.489
0 - val_loss: 1.0694 - val_acc: 0.5927
Epoch 15/50
26/26 [==============================] - 3s 127ms/step - loss: 1.4102 - acc: 0.477
5 - val_loss: 1.1161 - val_acc: 0.5844
Epoch 16/50
26/26 [==============================] - 3s 130ms/step - loss: 1.3461 - acc: 0.497
3 - val_loss: 1.1296 - val_acc: 0.5797
Epoch 17/50
26/26 [==============================] - 3s 131ms/step - loss: 1.3144 - acc: 0.506
9 - val_loss: 0.9783 - val_acc: 0.6163
Epoch 18/50
26/26 [==============================] - 4s 158ms/step - loss: 1.3539 - acc: 0.495
3 - val_loss: 0.9571 - val_acc: 0.6387
Epoch 19/50
26/26 [==============================] - 3s 127ms/step - loss: 1.2994 - acc: 0.511
2 - val_loss: 1.2124 - val_acc: 0.5124
Epoch 20/50
26/26 [==============================] - 3s 126ms/step - loss: 1.2686 - acc: 0.533
0 - val_loss: 0.9111 - val_acc: 0.6316
Epoch 21/50
26/26 [==============================] - 3s 127ms/step - loss: 1.2472 - acc: 0.536
5 - val_loss: 0.8396 - val_acc: 0.6671
Epoch 22/50
26/26 [==============================] - 3s 130ms/step - loss: 1.2557 - acc: 0.523
```

```
3 - val_loss: 1.0563 - val_acc: 0.6139
Epoch 23/50
26/26 [==============================] - 3s 127ms/step - loss: 1.2631 - acc: 0.523
2 - val_loss: 0.9161 - val_acc: 0.6564
Epoch 24/50
26/26 [==============================] - 3s 130ms/step - loss: 1.2036 - acc: 0.551
0 - val_loss: 0.8778 - val_acc: 0.6671
Epoch 25/50
26/26 [==============================] - 3s 132ms/step - loss: 1.1811 - acc: 0.565
0 - val_loss: 0.9633 - val_acc: 0.6328
Accuracy:  0.6328217237308147
Epoch 1/50
26/26 [==============================] - 5s 193ms/step - loss: 3.7746 - acc: 0.139
4 - val_loss: 2.1744 - val_acc: 0.1429
Epoch 2/50
26/26 [==============================] - 3s 124ms/step - loss: 2.1145 - acc: 0.204
8 - val_loss: 1.9359 - val_acc: 0.2609
Epoch 3/50
26/26 [==============================] - 3s 129ms/step - loss: 2.0687 - acc: 0.253
5 - val_loss: 1.9719 - val_acc: 0.2456
Epoch 4/50
26/26 [==============================] - 3s 126ms/step - loss: 1.9513 - acc: 0.285
8 - val_loss: 1.8135 - val_acc: 0.3129
Epoch 5/50
26/26 [==============================] - 3s 126ms/step - loss: 1.9184 - acc: 0.313
8 - val_loss: 1.6744 - val_acc: 0.4321
Epoch 6/50
26/26 [==============================] - 3s 126ms/step - loss: 1.8421 - acc: 0.337
8 - val_loss: 1.5371 - val_acc: 0.4262
Epoch 7/50
26/26 [==============================] - 3s 126ms/step - loss: 1.7488 - acc: 0.354
2 - val_loss: 1.4836 - val_acc: 0.4368
Epoch 8/50
26/26 [==============================] - 3s 126ms/step - loss: 1.6810 - acc: 0.374
5 - val_loss: 1.2741 - val_acc: 0.5171
Epoch 9/50
26/26 [==============================] - 3s 126ms/step - loss: 1.6444 - acc: 0.396
1 - val_loss: 1.3490 - val_acc: 0.5065
Epoch 10/50
26/26 [==============================] - 3s 127ms/step - loss: 1.6205 - acc: 0.407
6 - val_loss: 1.3832 - val_acc: 0.5089
Epoch 11/50
26/26 [==============================] - 3s 126ms/step - loss: 1.6017 - acc: 0.412
3 - val_loss: 1.4431 - val_acc: 0.4569
Epoch 12/50
26/26 [==============================] - 3s 126ms/step - loss: 1.5199 - acc: 0.451
2 - val_loss: 1.2439 - val_acc: 0.5478
Epoch 13/50
26/26 [==============================] - 3s 126ms/step - loss: 1.5688 - acc: 0.444
1 - val_loss: 1.1775 - val_acc: 0.5478
Epoch 14/50
26/26 [==============================] - 3s 126ms/step - loss: 1.4374 - acc: 0.453
0 - val_loss: 1.1249 - val_acc: 0.5525
Epoch 15/50
26/26 [==============================] - 3s 126ms/step - loss: 1.4431 - acc: 0.464
2 - val_loss: 1.0849 - val_acc: 0.5561
Epoch 16/50
26/26 [==============================] - 3s 126ms/step - loss: 1.3898 - acc: 0.462
1 - val_loss: 1.2095 - val_acc: 0.5466
Epoch 17/50
26/26 [==============================] - 3s 126ms/step - loss: 1.4313 - acc: 0.464
```

```
0 - val_loss: 1.1274 - val_acc: 0.5584
Epoch 18/50
26/26 [==============================] - 3s 130ms/step - loss: 1.3163 - acc: 0.504
3 - val_loss: 1.0291 - val_acc: 0.5856
Epoch 19/50
26/26 [==============================] - 4s 155ms/step - loss: 1.3575 - acc: 0.499
8 - val_loss: 1.6595 - val_acc: 0.4427
Epoch 20/50
26/26 [==============================] - 4s 137ms/step - loss: 1.3294 - acc: 0.510
1 - val_loss: 1.1316 - val_acc: 0.5596
Epoch 21/50
26/26 [==============================] - 3s 130ms/step - loss: 1.2846 - acc: 0.501
4 - val_loss: 0.9215 - val_acc: 0.6505
Epoch 22/50
26/26 [==============================] - 3s 127ms/step - loss: 1.3240 - acc: 0.496
1 - val_loss: 0.9055 - val_acc: 0.6316
Epoch 23/50
26/26 [==============================] - 3s 134ms/step - loss: 1.2367 - acc: 0.546
2 - val_loss: 0.9689 - val_acc: 0.6246
Epoch 24/50
26/26 [==============================] - 5s 200ms/step - loss: 1.2440 - acc: 0.525
7 - val_loss: 0.9521 - val_acc: 0.6210
Epoch 25/50
26/26 [==============================] - 4s 148ms/step - loss: 1.2278 - acc: 0.528
2 - val_loss: 0.9110 - val_acc: 0.6588
Epoch 26/50
26/26 [==============================] - 3s 131ms/step - loss: 1.1816 - acc: 0.552
6 - val_loss: 0.8124 - val_acc: 0.6564
Epoch 27/50
26/26 [==============================] - 4s 138ms/step - loss: 1.2133 - acc: 0.543
7 - val_loss: 1.3262 - val_acc: 0.5325
Epoch 28/50
26/26 [==============================] - 4s 135ms/step - loss: 1.1441 - acc: 0.565
8 - val_loss: 1.0731 - val_acc: 0.6057
Epoch 29/50
26/26 [==============================] - 5s 210ms/step - loss: 1.1483 - acc: 0.560
4 - val_loss: 0.7630 - val_acc: 0.6812
Epoch 30/50
26/26 [==============================] - 6s 230ms/step - loss: 1.1794 - acc: 0.545
4 - val_loss: 0.9103 - val_acc: 0.6576
Epoch 31/50
26/26 [==============================] - 4s 144ms/step - loss: 1.1236 - acc: 0.566
8 - val_loss: 0.8646 - val_acc: 0.6659
Epoch 32/50
26/26 [==============================] - 3s 131ms/step - loss: 1.1409 - acc: 0.559
6 - val_loss: 1.2644 - val_acc: 0.5407
Epoch 33/50
26/26 [==============================] - 3s 133ms/step - loss: 1.1015 - acc: 0.583
5 - val_loss: 0.9128 - val_acc: 0.6340
Accuracy:  0.6340023612750886
Epoch 1/50
26/26 [==============================] - 5s 177ms/step - loss: 3.8083 - acc: 0.137
3 - val_loss: 2.1043 - val_acc: 0.1688
Epoch 2/50
26/26 [==============================] - 3s 131ms/step - loss: 2.0966 - acc: 0.222
0 - val_loss: 1.8822 - val_acc: 0.2645
Epoch 3/50
26/26 [==============================] - 3s 131ms/step - loss: 2.0578 - acc: 0.256
7 - val_loss: 1.8051 - val_acc: 0.3447
Epoch 4/50
26/26 [==============================] - 4s 138ms/step - loss: 2.0025 - acc: 0.290
```

```
1 - val_loss: 1.6765 - val_acc: 0.4156
Epoch 5/50
26/26 [==============================] - 3s 134ms/step - loss: 1.8765 - acc: 0.309
1 - val_loss: 1.5505 - val_acc: 0.4616
Epoch 6/50
26/26 [==============================] - 3s 134ms/step - loss: 1.8845 - acc: 0.316
9 - val_loss: 1.5801 - val_acc: 0.4227
Epoch 7/50
26/26 [==============================] - 3s 133ms/step - loss: 1.8173 - acc: 0.353
0 - val_loss: 1.4581 - val_acc: 0.4368
Epoch 8/50
26/26 [==============================] - 3s 133ms/step - loss: 1.7231 - acc: 0.369
6 - val_loss: 1.4142 - val_acc: 0.4876
Epoch 9/50
26/26 [==============================] - 3s 133ms/step - loss: 1.6750 - acc: 0.387
4 - val_loss: 1.6630 - val_acc: 0.4309
Epoch 10/50
26/26 [==============================] - 3s 133ms/step - loss: 1.6054 - acc: 0.412
9 - val_loss: 1.5469 - val_acc: 0.4604
Epoch 11/50
26/26 [==============================] - 3s 133ms/step - loss: 1.5943 - acc: 0.412
9 - val_loss: 1.3326 - val_acc: 0.5183
Epoch 12/50
26/26 [==============================] - 3s 133ms/step - loss: 1.5253 - acc: 0.434
4 - val_loss: 1.8623 - val_acc: 0.3920
Epoch 13/50
26/26 [==============================] - 3s 133ms/step - loss: 1.4867 - acc: 0.453
2 - val_loss: 1.3428 - val_acc: 0.5195
Epoch 14/50
26/26 [==============================] - 3s 133ms/step - loss: 1.5001 - acc: 0.445
5 - val_loss: 1.1789 - val_acc: 0.5667
Epoch 15/50
26/26 [==============================] - 3s 133ms/step - loss: 1.3895 - acc: 0.468
9 - val_loss: 1.0348 - val_acc: 0.6045
Epoch 16/50
26/26 [==============================] - 3s 133ms/step - loss: 1.4210 - acc: 0.469
0 - val_loss: 1.1489 - val_acc: 0.5514
Epoch 17/50
26/26 [==============================] - 3s 133ms/step - loss: 1.3702 - acc: 0.491
9 - val_loss: 1.3943 - val_acc: 0.5254
Epoch 18/50
26/26 [==============================] - 3s 133ms/step - loss: 1.3653 - acc: 0.485
4 - val_loss: 1.1999 - val_acc: 0.5502
Epoch 19/50
26/26 [==============================] - 3s 133ms/step - loss: 1.3267 - acc: 0.496
3 - val_loss: 0.9443 - val_acc: 0.6470
Epoch 20/50
26/26 [==============================] - 3s 133ms/step - loss: 1.2867 - acc: 0.516
6 - val_loss: 1.2860 - val_acc: 0.5490
Epoch 21/50
26/26 [==============================] - 3s 133ms/step - loss: 1.2604 - acc: 0.521
6 - val_loss: 0.9825 - val_acc: 0.6175
Epoch 22/50
26/26 [==============================] - 3s 133ms/step - loss: 1.2540 - acc: 0.517
4 - val_loss: 1.0335 - val_acc: 0.6009
Epoch 23/50
26/26 [==============================] - 3s 133ms/step - loss: 1.2314 - acc: 0.535
2 - val_loss: 0.8734 - val_acc: 0.6494
Epoch 24/50
26/26 [==============================] - 3s 133ms/step - loss: 1.2228 - acc: 0.544
0 - val_loss: 0.8662 - val_acc: 0.6647
```

```
Epoch 25/50
26/26 [==============================] - 3s 133ms/step - loss: 1.2127 - acc: 0.540
0 - val_loss: 1.0453 - val_acc: 0.5738
Epoch 26/50
26/26 [==============================] - 3s 133ms/step - loss: 1.2031 - acc: 0.536
6 - val_loss: 0.9391 - val_acc: 0.6375
Epoch 27/50
26/26 [==============================] - 3s 133ms/step - loss: 1.1837 - acc: 0.550
8 - val_loss: 0.8412 - val_acc: 0.6682
Epoch 28/50
26/26 [==============================] - 4s 135ms/step - loss: 1.1566 - acc: 0.555
0 - val_loss: 0.8705 - val_acc: 0.6694
Epoch 29/50
26/26 [==============================] - 3s 133ms/step - loss: 1.1470 - acc: 0.564
1 - val_loss: 0.9674 - val_acc: 0.6352
Epoch 30/50
26/26 [==============================] - 3s 133ms/step - loss: 1.1447 - acc: 0.565
4 - val_loss: 1.2213 - val_acc: 0.5584
Epoch 31/50
26/26 [==============================] - 3s 133ms/step - loss: 1.0958 - acc: 0.574
5 - val_loss: 0.7960 - val_acc: 0.6753
Epoch 32/50
26/26 [==============================] - 3s 134ms/step - loss: 1.1262 - acc: 0.577
6 - val_loss: 0.7338 - val_acc: 0.6930
Epoch 33/50
26/26 [==============================] - 3s 133ms/step - loss: 1.0989 - acc: 0.582
5 - val_loss: 1.0208 - val_acc: 0.6080
Epoch 34/50
26/26 [==============================] - 3s 133ms/step - loss: 1.1015 - acc: 0.576
7 - val_loss: 0.9159 - val_acc: 0.6269
Epoch 35/50
26/26 [==============================] - 3s 133ms/step - loss: 1.0843 - acc: 0.591
5 - val_loss: 1.0048 - val_acc: 0.6187
Epoch 36/50
26/26 [==============================] - 3s 132ms/step - loss: 1.0743 - acc: 0.587
3 - val_loss: 1.0158 - val_acc: 0.6222
Accuracy:  0.6221959858323495
[0.5678866587957497, 0.5395513577331759, 0.6328217237308147, 0.6340023612750886,
0.6221959858323495]
```

In [16]:

```
t = 2.26 / np.sqrt(10)
e = (1-np.array(acc_scores3))-(1-np.array(acc_scores4))
stdtot =np.std(e)

dbar = np.mean(e)
print('model2 3X3 vs model2 2X2 acc range :',dbar-t*stdtot,dbar+t*stdtot)
```

```
model2 3X3 vs model2 2X2 acc range : -0.12504270631278086 -0.06008126062936779
```

Becasue the range is not include 0, so we can say that with 95% confident level, model2 3X3 and model2 2X2 are statisitcally different base on accuracy.

In [17]:

```
from statistics import mean
print('Average accuracy for model2 3X3 ',mean(acc_scores3))
print('Average accuracy for model2 2X2 ',mean(acc_scores4))
```

```
Average accuracy for model2 3X3  0.6935064935064935
Average accuracy for model2 2X2  0.6009445100354192
```

Base on my statistics comparision, there is different between model2 3X3 and model2 2X2, and 3X3 have higher average accuracy score. So model2 3X3 is better.

In [16]:

```
t = 2.26 / np.sqrt(10)
e = (1-np.array(acc_scores1))-(1-np.array(acc_scores3))
stdtot =np.std(e)

dbar = np.mean(e)
print('model1 3X3 vs model2 3X3 acc range :',dbar-t*stdtot,dbar+t*stdtot)
```

```
model1 3X3 vs model2 3X3 acc range : 0.040028185409468404 0.10778763513362483
```

Becasue the range is not include 0, so we can say that with 95% confident level, model1 3X3 and model2 3X3 are statisitcally different base on accuracy.

In [17]:

```
from statistics import mean
print('Average accuracy for model1 3X3 ',mean(acc_scores1))
print('Average accuracy for model2 3X3 ',mean(acc_scores3))
```

```
Average accuracy for model1 3X3  0.6250295159386069
Average accuracy for model2 3X3  0.6989374262101534
```

model2 3X3 is better.

Base on my statistics comparision, there is different between model1 3X3 and model2 3X3, and model2 3X3 have higher average accuracy score. So model2 3X3 is better.

MLP

In [136]:

```
from sklearn import metrics as mt
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline

def compare_mlp_cnn(cnn, mlp, X_test, y_test):
    plt.figure(figsize=(15,5))
    if cnn is not None:
        yhat_cnn = np.argmax(cnn.predict(np.expand_dims(X_test, axis=1)), axis=1)
        acc_cnn = mt.accuracy_score(y_test,yhat_cnn)
        plt.subplot(1,2,1)
        cm = mt.confusion_matrix(y_test,yhat_cnn)
        cm = cm/np.sum(cm,axis=1)[:,np.newaxis]
        sns.heatmap(cm, annot=True, fmt='.2f')
        plt.title('CNN: '+str(acc_cnn))

    if mlp is not None:
        yhat_mlp = np.argmax(mlp.predict(X_test), axis=1)
        acc_mlp = mt.accuracy_score(y_test,yhat_mlp)
        plt.subplot(1,2,2)
        cm = mt.confusion_matrix(y_test,yhat_mlp)
        cm = cm/np.sum(cm,axis=1)[:,np.newaxis]
        sns.heatmap(cm,annot=True, fmt='.2f')
        plt.title('MLP: '+str(acc_mlp))
```

In [155]:

```
#create the array contain the images data with 1-D image features
X = []
y = []
#seprate the 1-d images data and column values
for features, label in images:
    X.append(features)
    y.append(label)

names = y.copy()
#reshape the array into the correct format, which each row represent one images
X_images = np.array(X).reshape(-1,imagesize,imagesize)
X = np.array(X_images).reshape(len(images),imagesize*imagesize)
X = cv2.normalize(X, None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
_, h, img_wh = X_images.shape

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
y_train_ohe = keras.utils.to_categorical(y_train, NUM_CLASSES)
y_test_ohe = keras.utils.to_categorical(y_test, NUM_CLASSES)
```

In [52]:

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(50,),
                       solver='sgd',
                       learning_rate_init=0.01,
                       max_iter=100,
                       random_state=1)
```

In [54]:

```
%%time
mlp.fit(X_train, y_train_ohe)
#accuracy score
yhat = mlp.predict(X_test)
#https://stackoverflow.com/questions/46953967/multilabel-indicator-is-not-supported-for-confusio
n-matrix
print('Accuracy ',mt.accuracy_score(y_test_ohe.argmax(axis=1),yhat.argmax(axis=1)))
```

```
Accuracy  0.10153482880755609
Wall time: 2min 7s

D:\APP\conda\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:56
2: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

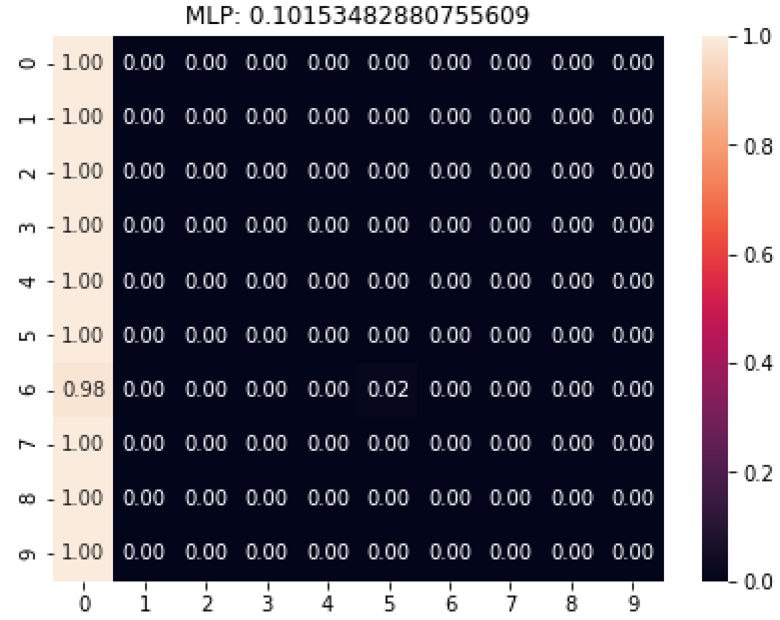
In [30]:

```
acc_scores5 = []


skf5 = StratifiedKFold(n_splits=num_folds, shuffle=True)
for i, (train, test) in enumerate(skf5.split(X, y)):

    mlp.fit(X_train, y_train_ohe)

    yhat = mlp.predict(X_test)
    acc_score5 = mt.accuracy_score(y_test_ohe.argmax(axis=1),yhat.argmax(axis=1))
    acc_scores5.append(acc_score5)#append all acc for k-fold
    print("Accuracy: ", acc_score5)

print(acc_scores5)
```

```
Accuracy:  0.08382526564344746
Accuracy:  0.08382526564344746
Accuracy:  0.08382526564344746
Accuracy:  0.08382526564344746
Accuracy:  0.08382526564344746
[0.08382526564344746, 0.08382526564344746, 0.08382526564344746, 0.0838252656434474
6, 0.08382526564344746]
```

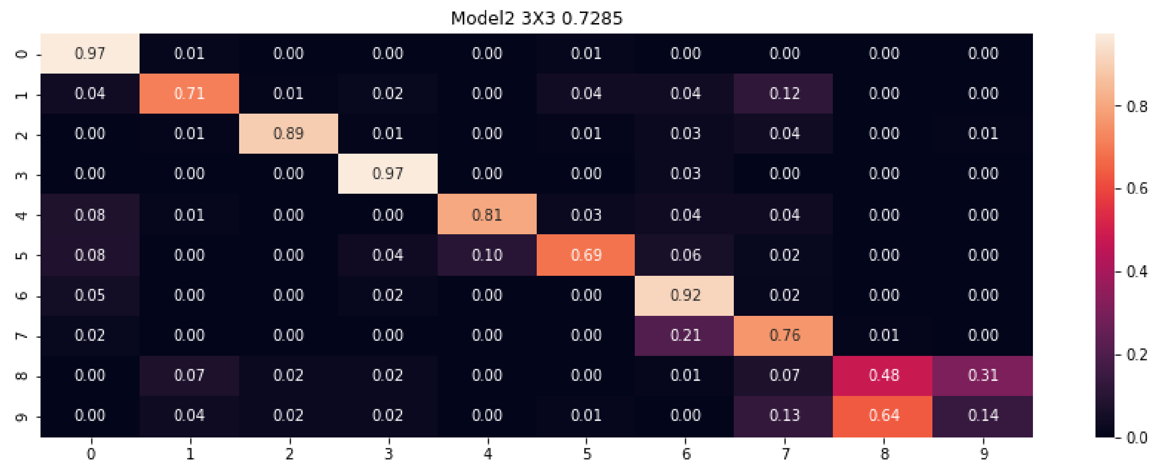
In [55]:

```
compare_mlp_cnn(None,mlp,X_test,y_test)
```



In [71]:

```
summarize_net(cnn3, X_test, y_test, title_text='Model2 3X3 ')
```



In [134]:

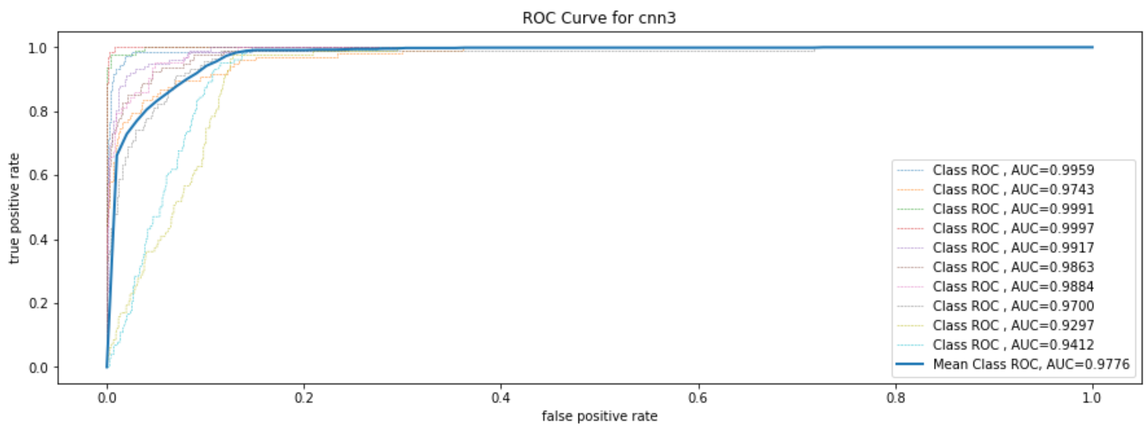
```
#https://github.com/egabrielsen/MachineLearning/blob/master/Lab08/lab8-Danh-Copy1.ipynb
#https://github.com/Nchaos/CNNs/blob/master/notebooks/Lab6_NC_CNNs.ipynb
#also modified from Dr.Larson 09 .Evaluation
def plot_roc(probas, y_true):
    plt.figure(figsize=(15,5))
    mean_tpr = 0.0
    mean_fpr = np.linspace(0, 1, 100)
    all_tpr = []

    classes = np.unique(y)#i have 10 class in total
    perclass_mean_tpr = 0.0
    roc_auc = 0
    for j in classes:
        fpr, tpr, thresholds = mt.roc_curve(y_test_ohe.argmax(axis=1), probas[:, j], pos_label=j
)#generate roc curve
        perclass_mean_tpr += interp(mean_fpr, fpr, tpr)
        perclass_mean_tpr[0] = 0.0
        roc_auc += mt.auc(fpr, tpr)
        #all above just modified of Dr.Larson code
        #here plot the roc curve for each class
        plt.plot(fpr,tpr,'--',lw=.5,label='Class ROC , AUC=%0.4f' %(mt.auc(fpr, tpr)))

    perclass_mean_tpr /= len(classes)
    roc_auc /= len(classes)
    mean_tpr += perclass_mean_tpr
    #this curve is the mean roc for all class
    plt.plot(mean_fpr,perclass_mean_tpr,'-',lw=2,label='Mean Class ROC, AUC=%0.4f'
                 %(roc_auc))
    plt.legend(loc='best')
    plt.xlabel('false positive rate')
    plt.ylabel('true positive rate')
```

In [135]:

```
probas = cnn3.predict_proba(X_test)
plot_roc(probas, y_test_ohe)
plt.title('ROC Curve for cnn3')
```

Out[135]:

Text(0.5, 1.0, 'ROC Curve for cnn3')

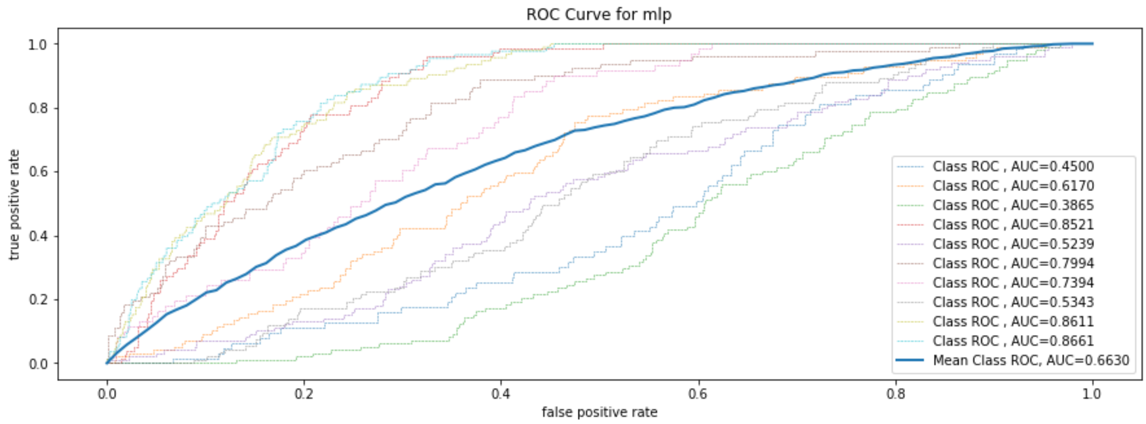


In [138]:

```
probas = mlp.predict_proba(X_test)
plot_roc(probas, y_test_ohe)
plt.title('ROC Curve for mlp')
```

Out[138]:

```
Text(0.5, 1.0, 'ROC Curve for mlp')
```



In [130]:

```
t = 2.26 / np.sqrt(10)
e = (1-np.array(acc_scores3))-(1-np.array(acc_scores5))
stdtot =np.std(e)

dbar = np.mean(e)
print('model2 3X3 vs mlp acc range :',dbar-t*stdtot,dbar+t*stdtot)
```

```
model2 3X3 vs mlp acc range : -0.6430612670487764 -0.5871630540846354
```

Becasue the range is not include 0, so we can say that with 95% confident level, model2 3X3 and mlp are statisitcally different base on accuracy.

In [139]:

```
from statistics import mean
print('Average accuracy for model1 3X3 ',mean(acc_scores3))
print('Average accuracy for mlp ',mean(acc_scores5))
```

```
Average accuracy for model1 3X3  0.6989374262101534
Average accuracy for mlp  0.08382526564344746
```

From above two ROC curve and AUC, it is clear to see that cnn model is better. It have better curve, with thresholds that false positive rate low and keep true positive rate high. Also the AUC is much more better than mlp one. Also the accuracy is way better than mlp.

Exceptional Work - Transfer Learning With ResNet

In [219]:

```
# manipulated from Keras Documentation
#  https://keras.io/applications/
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np
```

In [220]:

```
images=[] #just reimport the data from the files and resize the images as well
def creat_data():
    for i in Category:
        path = os.path.join(datadir, i)
        class_num = Category.index(i) #set a index for the column which will help me to set the
 names for different leaves
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_COLOR)
            resize_array = cv2.resize(img_array, (imagesize,imagesize))
            images.append([resize_array, class_num])# append the 1-d images data and the column
 index

creat_data()
```

In [221]:

```
from keras.datasets import cifar10
from keras.utils import to_categorical

NUM_CLASSES = 10
#create the array contain the images data with 1-D image features
X = []
y = []
#seprate the 1-d images data and column values
for features, label in images:
    X.append(features)
    y.append(label)
y = np.array(y)
names = y.copy()
#reshape the array into the correct format, which each row represent one images
X_images = np.array(X).reshape(-1,imagesize,imagesize)
X = np.array(X_images).reshape(len(images),imagesize,imagesize,3)
X = cv2.normalize(X, None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
_, h, img_wh = X_images.shape

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
# NEW: Let's start by fixing the sizes


y_train_ohe = to_categorical(y_train, NUM_CLASSES)
y_test_ohe = to_categorical(y_test, NUM_CLASSES)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(3384, 150, 150, 3)
(3384,)
(847, 150, 150, 3)
(847,)
```

In [222]:

```
# from skimage.transform import resize # stupid slow function
from scipy.misc import imresize
import numpy as np


X_train_up = [imresize(X, size=(150,150,3), interp='nearest') for X in X_train]
X_train_up = np.stack(X_train_up, axis=0)
print(X_train_up.shape)

X_test_up = [imresize(X, size=(150,150,3), interp='nearest') for X in X_test]
X_test_up = np.stack(X_test_up, axis=0)
print(X_test_up.shape)
```

```
D:\APP\conda\lib\site-packages\ipykernel_launcher.py:7: DeprecationWarning: `imres
ize` is deprecated!
`imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.3.0.
Use Pillow instead: ``numpy.array(Image.fromarray(arr).resize())``.
  import sys
```

```
(3384, 150, 150, 3)
```

```
D:\APP\conda\lib\site-packages\ipykernel_launcher.py:11: DeprecationWarning: `imre
size` is deprecated!
`imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.3.0.
Use Pillow instead: ``numpy.array(Image.fromarray(arr).resize())``.
  # This is added back by InteractiveShellApp.init_path()
```

```
(847, 150, 150, 3)
```

In [224]:

```
# connect new layers to the output
from keras.applications.resnet50 import ResNet50
from keras.applications.resnet50 import preprocess_input, decode_predictions

# load only convolutional layers of resnet:
if 'res_no_top' not in locals():
    res_no_top = ResNet50(weights='imagenet', include_top=False)

X = X_train_up[0]
X = np.expand_dims(X, axis=0)
X = preprocess_input(X)

%time preds = res_no_top.predict(X)
preds.shape
```

```
Wall time: 19.5 ms
```

Out[224]:

```
(1, 5, 5, 2048)
```

In [225]:

```
X_train_up = preprocess_input(X_train_up)
X_test_up = preprocess_input(X_test_up)
```

In [226]:

```
# train on half the data, to save a few hours
X_train_resnet = res_no_top.predict(X_train_up)
X_test_resnet = res_no_top.predict(X_test_up)
print(X_train_resnet.shape)
```

```
(3384, 5, 5, 2048)
```

In [227]:

```
from keras.layers import SeparableConv2D
from keras.layers.normalization import BatchNormalization
from keras.layers import Add, Flatten, Dense
from keras.layers import average, concatenate
from keras.models import Input, Model


# let's add a fully-connected layer
input_X = Input(shape=X_train_resnet[0].shape)
X = Flatten()(input_X)
x = Dense(200, activation='relu',kernel_initializer='he_uniform')(X)
# and a fully connected layer
predictions = Dense(NUM_CLASSES, activation='softmax', kernel_initializer='glorot_uniform')(X)

model = Model(inputs=input_X, outputs=predictions)

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         (None, 5, 5, 2048)        0
_________________________________________________________________
flatten_12 (Flatten)         (None, 51200)             0
_________________________________________________________________
dense_24 (Dense)             (None, 10)                512010
=================================================================
Total params: 512,010
Trainable params: 512,010
Non-trainable params: 0
_________________________________________________________________
```

In [228]:

```
y_train_ohe_resnet = y_train_ohe[:X_train_resnet.shape[0]]

model.compile(optimizer='rmsprop', loss='categorical_crossentropy',metrics=['accuracy'])

model.fit(X_train_resnet,y_train_ohe_resnet,epochs=10,
          batch_size=64,verbose=1,
         validation_data=(X_test_resnet, y_test_ohe[:X_test_resnet.shape[0]]))
```

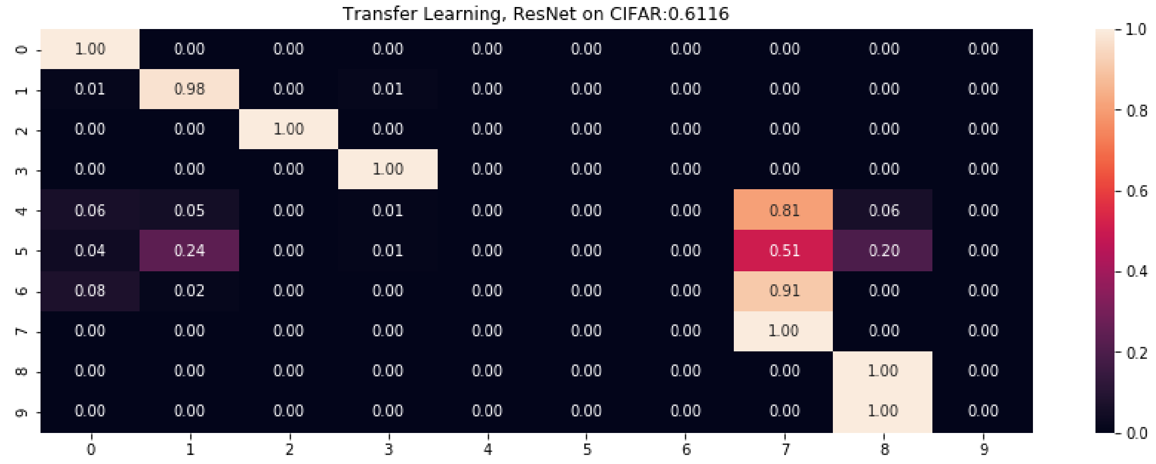
```
Train on 3384 samples, validate on 847 samples
Epoch 1/10
3384/3384 [==============================] - 2s 626us/step - loss: 8.5675 - acc:
0.4362 - val_loss: 8.0371 - val_acc: 0.4970
Epoch 2/10
3384/3384 [==============================] - 1s 195us/step - loss: 7.5472 - acc:
0.5245 - val_loss: 6.3493 - val_acc: 0.5927
Epoch 3/10
3384/3384 [==============================] - 1s 194us/step - loss: 6.4681 - acc:
0.5919 - val_loss: 6.2771 - val_acc: 0.6068
Epoch 4/10
3384/3384 [==============================] - 1s 194us/step - loss: 6.3949 - acc:
0.5996 - val_loss: 6.3211 - val_acc: 0.6068
Epoch 5/10
3384/3384 [==============================] - 1s 194us/step - loss: 6.3707 - acc:
0.6040 - val_loss: 6.2491 - val_acc: 0.6116
Epoch 6/10
3384/3384 [==============================] - 1s 194us/step - loss: 6.3750 - acc:
0.6034 - val_loss: 6.3838 - val_acc: 0.5998
Epoch 7/10
3384/3384 [==============================] - 1s 193us/step - loss: 6.3578 - acc:
0.6052 - val_loss: 6.2426 - val_acc: 0.6116
Epoch 8/10
3384/3384 [==============================] - 1s 197us/step - loss: 6.3539 - acc:
0.6058 - val_loss: 6.2422 - val_acc: 0.6116
Epoch 9/10
3384/3384 [==============================] - 1s 194us/step - loss: 6.3539 - acc:
0.6058 - val_loss: 6.2414 - val_acc: 0.6116
Epoch 10/10
3384/3384 [==============================] - 1s 194us/step - loss: 6.3539 - acc:
0.6058 - val_loss: 6.2410 - val_acc: 0.6116
```

Out[228]:

```
<keras.callbacks.History at 0x2454321ce80>
```

In [229]:

```
summarize_net(model, X_test_resnet, y_test[:X_test_resnet.shape[0]], title_text='Transfer Learning, ResNet on CIFAR:')
```



Comparing to my best model Model2 3X3, the accuracy is 0.72, which is higher than Transfer Learning With ResNet(0.61). So my model is better base on accuracy score.

In [ ]:

```

```