In [1]:
```
# import libraries and magics
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from PIL import Image
import cv2

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

import tensorflow as tf
from tensorflow import keras
```

Problem 1

In [2]:
```
# Loading Training Data
X_train_full = np.load('flower_species_classification/data_train.npy').T
t_train_full = np.load('flower_species_classification/labels_train.npy')

class_names = ['Roses', 'Magnolias', 'Lilies', 'Sunflowers', 'Orchids',
               'Marigold', 'Hibiscus', 'Firebush', 'Pentas', 'Bougainvillea']

# X_val, X_train = X_train_full[:300]/255.0, X_train_full[300:]/255.0
# t_val, t_train = t_train_full[:300], t_train_full[300:]

X_train, X_val, t_train, t_val = train_test_split(X_train_full, t_train_full,
                                                  test_size=0.2,
                                                  stratify=t_train_full,
                                                  shuffle=True,
                                                  random_state=42)

print(X_train_full.shape, t_train_full.shape)
print(X_train.shape, t_train.shape)
print(X_val.shape, t_val.shape)
```

```
(1658, 270000) (1658,)
(1326, 270000) (1326,)
(332, 270000) (332,)
```

In [3]:
```
# Scale the training data
X_train_scaled = X_train / 255.0
X_val_scaled = X_val /255.0
```

In [5]:
```
X_train_rs = tf.constant(X_train_scaled.reshape((X_train_scaled.shape[0],300,300,3)),
                         dtype=tf.float16)
X_val_rs = tf.constant(X_val_scaled.reshape((X_val_scaled.shape[0], 300,300,3)),
                       dtype=tf.float16)

X_train_rs.shape, X_val_rs.shape
```

Out[5]:
```
(TensorShape([1326, 300, 300, 3]), TensorShape([332, 300, 300, 3]))
```

In [6]:
```
# Define function for evaluating performance
def Evaluate_performance(model, history, Name, X_train=X_train_rs, t_train=t_train,

    y_train = np.argmax(model.predict(X_train),axis=1)
    y_val = np.argmax(model.predict(X_val),axis=1)

    # Accuracy
```

```
    train_acc = accuracy_score(y_train, t_train)
    val_acc = accuracy_score(y_val, t_val)

    # Print performance
    print('Performance of {}:\n'.format(Name))
    print('1. In training set: ')
    print(classification_report(t_train, y_train))
    print('Accuracy: {}'.format(train_acc))
    print('Confusion Matrix')
    print(confusion_matrix(t_train, y_train))

    print('\n ===================================================== \n')

    print('2. In validation set: ')
    print(classification_report(t_val, y_val))
    print('Accuracy: {}'.format(val_acc))
    print('Confusion Matrix')
    print(confusion_matrix(t_val, y_val))

    # Display learning curve
    if display==True:
        key_names = list(history.history.keys())
        colors = ['-r','--b','-og','-.k']

        plt.figure(figsize=(8,5))
        for i in [0,2]:
            plt.plot(history.history[key_names[i]], colors[i], label=key_names[i])
        plt.legend(fontsize=15,ncol=2)
        plt.title('Learning Curves with loss', size=15);

        plt.figure(figsize=(8,5))
        for i in [1,3]:
            plt.plot(history.history[key_names[i]], colors[i], label=key_names[i])
        plt.legend(fontsize=15,ncol=2)
        plt.title('Learning Curves with accuracy', size=15);
```

In [7]:

```
# Model 1: Only use ANN
model_prob1_1 = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[300,300,3]),
    keras.layers.Dense(300, kernel_initializer='he_normal'),
    keras.layers.LeakyReLU(alpha=0.2),
    keras.layers.Dense(100, kernel_initializer='he_normal'),
    keras.layers.LeakyReLU(alpha=0.2),
    keras.layers.Dense(10, activation='softmax')
    ])
```

In [8]:

```
model_prob1_1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 flatten (Flatten)           (None, 270000)            0         
                                                                 
 dense (Dense)               (None, 300)               81000300  
                                                                 
 leaky_re_lu (LeakyReLU)     (None, 300)               0         
                                                                 
 dense_1 (Dense)             (None, 100)               30100     
                                                                 
 leaky_re_lu_1 (LeakyReLU)   (None, 100)               0         
                                                                 
 dense_2 (Dense)             (None, 10)                1010      
                                                                 
=================================================================
Total params: 81,031,410
Trainable params: 81,031,410
Non-trainable params: 0
_________________________________________________________________
```

In [9]:
```
model_prob1_1.compile(optimizer=keras.optimizers.Nadam(),
                      loss=keras.losses.SparseCategoricalCrossentropy(),
                      metrics=['accuracy'])
```

In [11]:
```
earlystop = keras.callbacks.EarlyStopping(monitor='val_loss',
                                          patience=20)
checkpoint = keras.callbacks.ModelCheckpoint('Model/model_problem1_one.h5',
                                             save_best_only=True)

history = model_prob1_1.fit(X_train_rs, t_train,
                            epochs=100,
                            batch_size=32,
                            validation_data=(X_val_rs, t_val),
                            callbacks=[earlystop, checkpoint])
```

```
Epoch 1/100
42/42 [==============================] - 2s 49ms/step - loss: 20.6611 - accuracy: 0.
2722 - val_loss: 13.3653 - val_accuracy: 0.2289
Epoch 2/100
42/42 [==============================] - 2s 49ms/step - loss: 22.1139 - accuracy: 0.
2896 - val_loss: 9.1887 - val_accuracy: 0.3916
Epoch 3/100
42/42 [==============================] - 1s 17ms/step - loss: 5.6325 - accuracy: 0.4
744 - val_loss: 10.3024 - val_accuracy: 0.2108
Epoch 4/100
42/42 [==============================] - 1s 18ms/step - loss: 5.7164 - accuracy: 0.4
698 - val_loss: 13.6342 - val_accuracy: 0.2952
Epoch 5/100
42/42 [==============================] - 1s 17ms/step - loss: 32.5836 - accuracy: 0.
2602 - val_loss: 18.2549 - val_accuracy: 0.3494
Epoch 6/100
42/42 [==============================] - 2s 51ms/step - loss: 7.5491 - accuracy: 0.5
030 - val_loss: 8.3327 - val_accuracy: 0.3193
Epoch 7/100
42/42 [==============================] - 2s 51ms/step - loss: 3.8687 - accuracy: 0.5
686 - val_loss: 4.6841 - val_accuracy: 0.3735
Epoch 8/100
42/42 [==============================] - 1s 18ms/step - loss: 3.0509 - accuracy: 0.6
192 - val_loss: 6.4630 - val_accuracy: 0.4036
Epoch 9/100
42/42 [==============================] - 1s 17ms/step - loss: 2.2315 - accuracy: 0.6
425 - val_loss: 6.1322 - val_accuracy: 0.3735
Epoch 10/100
42/42 [==============================] - 1s 18ms/step - loss: 1.8034 - accuracy: 0.6
704 - val_loss: 6.4553 - val_accuracy: 0.3012
Epoch 11/100
42/42 [==============================] - 1s 17ms/step - loss: 1.3972 - accuracy: 0.7
511 - val_loss: 5.8732 - val_accuracy: 0.3223
Epoch 12/100
42/42 [==============================] - 1s 17ms/step - loss: 1.4681 - accuracy: 0.7
323 - val_loss: 5.8218 - val_accuracy: 0.3524
Epoch 13/100
42/42 [==============================] - 2s 49ms/step - loss: 0.9119 - accuracy: 0.8
047 - val_loss: 2.7044 - val_accuracy: 0.5120
Epoch 14/100
42/42 [==============================] - 1s 17ms/step - loss: 0.2885 - accuracy: 0.9
140 - val_loss: 4.0227 - val_accuracy: 0.4157
Epoch 15/100
42/42 [==============================] - 2s 52ms/step - loss: 0.6597 - accuracy: 0.8
575 - val_loss: 2.5413 - val_accuracy: 0.5211
Epoch 16/100
42/42 [==============================] - 1s 17ms/step - loss: 0.1630 - accuracy: 0.9
502 - val_loss: 3.1840 - val_accuracy: 0.4217
Epoch 17/100
42/42 [==============================] - 1s 17ms/step - loss: 0.2323 - accuracy: 0.9
314 - val_loss: 2.6710 - val_accuracy: 0.5000
Epoch 18/100
42/42 [==============================] - 2s 50ms/step - loss: 0.2112 - accuracy: 0.9
510 - val_loss: 2.3693 - val_accuracy: 0.5211
Epoch 19/100
42/42 [==============================] - 2s 52ms/step - loss: 0.0399 - accuracy: 0.9
947 - val_loss: 2.3343 - val_accuracy: 0.5301
Epoch 20/100
42/42 [==============================] - 2s 52ms/step - loss: 0.0265 - accuracy: 0.9
970 - val_loss: 2.3103 - val_accuracy: 0.5301
Epoch 21/100
42/42 [==============================] - 2s 50ms/step - loss: 0.0189 - accuracy: 1.0
000 - val_loss: 2.2675 - val_accuracy: 0.5392
Epoch 22/100
```

```
42/42 [==============================] - 1s 17ms/step - loss: 0.0142 - accuracy: 1.0
000 - val_loss: 2.2933 - val_accuracy: 0.5271
Epoch 23/100
42/42 [==============================] - 1s 17ms/step - loss: 0.0128 - accuracy: 1.0
000 - val_loss: 2.2774 - val_accuracy: 0.5241
Epoch 24/100
42/42 [==============================] - 1s 17ms/step - loss: 0.0113 - accuracy: 1.0
000 - val_loss: 2.2773 - val_accuracy: 0.5301
Epoch 25/100
42/42 [==============================] - 1s 16ms/step - loss: 0.0096 - accuracy: 1.0
000 - val_loss: 2.2915 - val_accuracy: 0.5301
Epoch 26/100
42/42 [==============================] - 1s 16ms/step - loss: 0.0093 - accuracy: 1.0
000 - val_loss: 2.3352 - val_accuracy: 0.5211
Epoch 27/100
42/42 [==============================] - 1s 18ms/step - loss: 0.0083 - accuracy: 1.0
000 - val_loss: 2.3242 - val_accuracy: 0.5271
Epoch 28/100
42/42 [==============================] - 1s 17ms/step - loss: 0.0076 - accuracy: 1.0
000 - val_loss: 2.3424 - val_accuracy: 0.5331
Epoch 29/100
42/42 [==============================] - 1s 18ms/step - loss: 0.0071 - accuracy: 1.0
000 - val_loss: 2.4101 - val_accuracy: 0.5211
Epoch 30/100
42/42 [==============================] - 1s 18ms/step - loss: 0.0067 - accuracy: 1.0
000 - val_loss: 2.3661 - val_accuracy: 0.5271
Epoch 31/100
42/42 [==============================] - 1s 18ms/step - loss: 0.0060 - accuracy: 1.0
000 - val_loss: 2.3848 - val_accuracy: 0.5422
Epoch 32/100
42/42 [==============================] - 1s 18ms/step - loss: 0.0054 - accuracy: 1.0
000 - val_loss: 2.4165 - val_accuracy: 0.5361
Epoch 33/100
42/42 [==============================] - 1s 17ms/step - loss: 0.0051 - accuracy: 1.0
000 - val_loss: 2.4576 - val_accuracy: 0.5301
Epoch 34/100
42/42 [==============================] - 1s 19ms/step - loss: 0.0045 - accuracy: 1.0
000 - val_loss: 2.4362 - val_accuracy: 0.5301
Epoch 35/100
42/42 [==============================] - 1s 17ms/step - loss: 0.0041 - accuracy: 1.0
000 - val_loss: 2.5018 - val_accuracy: 0.5271
Epoch 36/100
42/42 [==============================] - 1s 16ms/step - loss: 0.0037 - accuracy: 1.0
000 - val_loss: 2.5313 - val_accuracy: 0.5331
Epoch 37/100
42/42 [==============================] - 1s 17ms/step - loss: 0.0032 - accuracy: 1.0
000 - val_loss: 2.5721 - val_accuracy: 0.5331
Epoch 38/100
42/42 [==============================] - 1s 17ms/step - loss: 0.0028 - accuracy: 1.0
000 - val_loss: 2.5916 - val_accuracy: 0.5301
Epoch 39/100
42/42 [==============================] - 1s 18ms/step - loss: 0.0024 - accuracy: 1.0
000 - val_loss: 2.6498 - val_accuracy: 0.5301
Epoch 40/100
42/42 [==============================] - 1s 18ms/step - loss: 0.0022 - accuracy: 1.0
000 - val_loss: 2.6370 - val_accuracy: 0.5301
Epoch 41/100
42/42 [==============================] - 1s 19ms/step - loss: 0.0020 - accuracy: 1.0
000 - val_loss: 2.6972 - val_accuracy: 0.5271
```

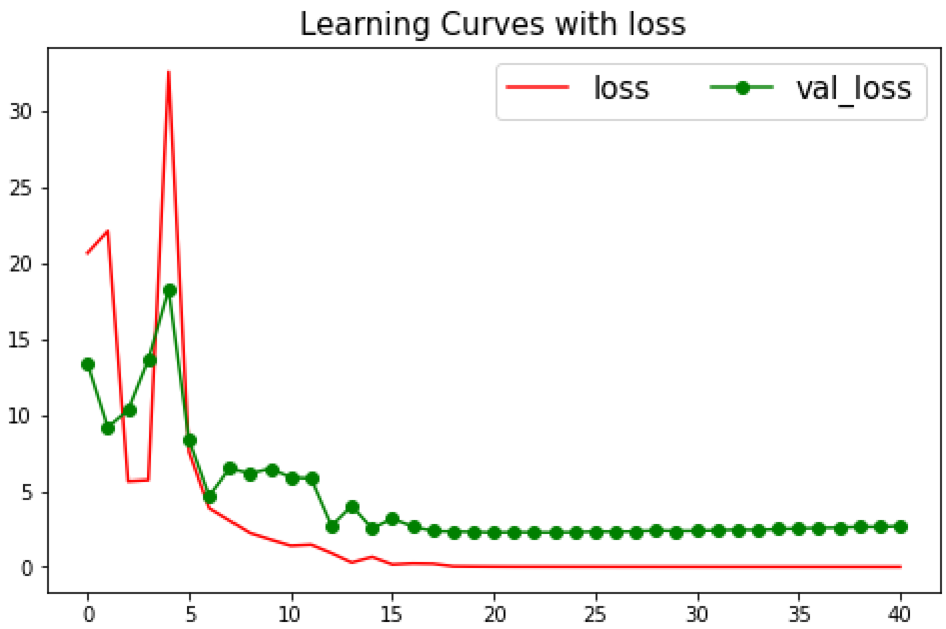
In [12]:

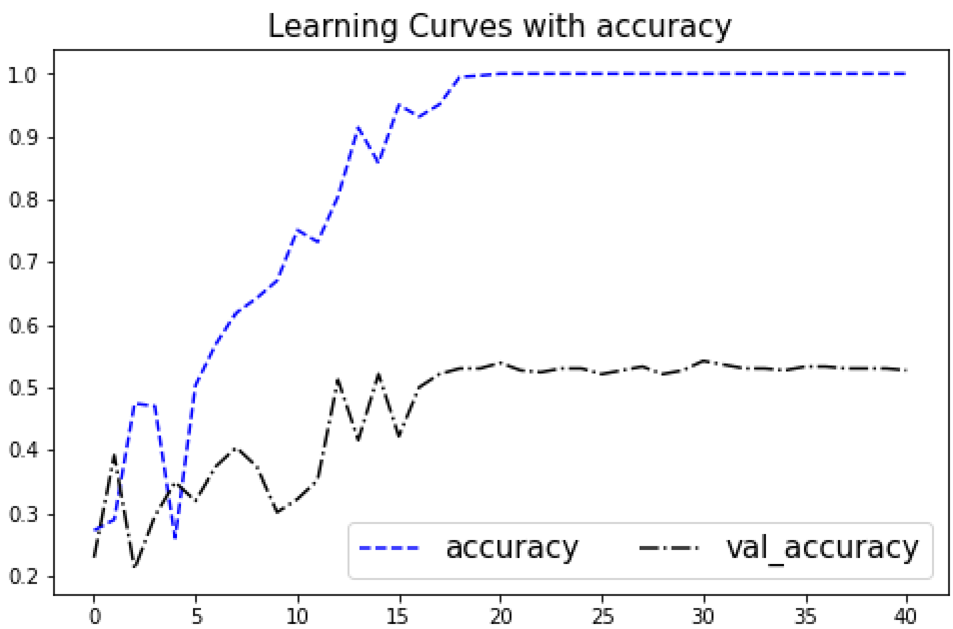
```
## Performance result
model = keras.models.load_model('Model/model_problem1_one.h5')
Evaluate_performance(model=model, history=history, Name='ANN')
```

Performance of ANN:

```
1. In training set:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       141
         1.0       1.00      1.00      1.00       144
         2.0       1.00      1.00      1.00       164
         3.0       1.00      1.00      1.00       112
         4.0       1.00      1.00      1.00       138
         5.0       1.00      1.00      1.00       125
         6.0       1.00      1.00      1.00       128
         7.0       1.00      1.00      1.00       138
         8.0       1.00      1.00      1.00       130
         9.0       1.00      1.00      1.00       106

    accuracy                           1.00      1326
   macro avg       1.00      1.00      1.00      1326
weighted avg       1.00      1.00      1.00      1326

Accuracy: 1.0
Confusion Matrix
[[141   0   0   0   0   0   0   0   0   0]
 [  0 144   0   0   0   0   0   0   0   0]
 [  0   0 164   0   0   0   0   0   0   0]
 [  0   0   0 112   0   0   0   0   0   0]
 [  0   0   0   0 138   0   0   0   0   0]
 [  0   0   0   0   0 125   0   0   0   0]
 [  0   0   0   0   0   0 128   0   0   0]
 [  0   0   0   0   0   0   0 138   0   0]
 [  0   0   0   0   0   0   0   0 130   0]
 [  0   0   0   0   0   0   0   0   0 106]]

 =====================================================

2. In validation set:
              precision    recall  f1-score   support

         0.0       0.51      0.50      0.51        36
         1.0       0.68      0.75      0.71        36
         2.0       0.32      0.39      0.35        41
         3.0       0.60      0.64      0.62        28
         4.0       0.67      0.63      0.65        35
         5.0       0.66      0.68      0.67        31
         6.0       0.57      0.41      0.47        32
         7.0       0.64      0.68      0.66        34
         8.0       0.41      0.47      0.43        32
         9.0       0.38      0.22      0.28        27

    accuracy                           0.54       332
   macro avg       0.54      0.54      0.53       332
weighted avg       0.54      0.54      0.54       332

Accuracy: 0.5391566265060241
Confusion Matrix
[[18  2  3  0  3  0  4  2  1  3]
 [ 0 27  4  0  3  0  0  1  1  0]
 [ 2  4 16  5  3  1  2  1  4  3]
 [ 0  0  3 18  0  4  0  3  0  0]
 [ 0  4  5  0 22  1  0  0  2  1]
 [ 0  0  2  5  0 21  0  2  1  0]
 [ 4  2  4  0  0  3 13  1  4  1]
 [ 3  0  3  1  0  2  1 23  1  0]
```

```
[ 3  1  5  0  0  0  3  3 15  2]
[ 5  0  5  1  2  0  0  0  8  6]]
```





In [22]:
```
# Model 2: With convolution layers
model_prob1_2 = keras.models.Sequential([
    keras.layers.Conv2D(64, 10, activation='selu', padding='same', input_shape=[300,
    keras.layers.MaxPooling2D(2),
    keras.layers.Conv2D(128, 5, activation='selu', padding='same', kernel_initialize
    keras.layers.MaxPooling2D(2),
    keras.layers.Conv2D(256, 3, activation='selu', padding='same', kernel_initialize
    keras.layers.MaxPooling2D(2),
    keras.layers.Flatten(),
    keras.layers.Dense(300, kernel_initializer='he_normal'),
    keras.layers.LeakyReLU(alpha=0.2),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(100, kernel_initializer='he_normal'),
    keras.layers.LeakyReLU(alpha=0.2),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(10, activation='softmax')
    ])
```

In [23]:
```
model_prob1_2.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_6 (Conv2D)           (None, 300, 300, 64)      19264     
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 150, 150, 64)     0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 150, 150, 128)     204928    
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 75, 75, 128)      0         
 2D)                                                             
                                                                 
 conv2d_8 (Conv2D)           (None, 75, 75, 256)       295168    
                                                                 
 max_pooling2d_8 (MaxPooling  (None, 37, 37, 256)      0         
 2D)                                                             
                                                                 
 flatten_3 (Flatten)         (None, 350464)            0         
                                                                 
 dense_9 (Dense)             (None, 300)               105139500 
                                                                 
 leaky_re_lu_6 (LeakyReLU)   (None, 300)               0         
                                                                 
 dropout_4 (Dropout)         (None, 300)               0         
                                                                 
 dense_10 (Dense)            (None, 100)               30100     
                                                                 
 leaky_re_lu_7 (LeakyReLU)   (None, 100)               0         
                                                                 
 dropout_5 (Dropout)         (None, 100)               0         
                                                                 
 dense_11 (Dense)            (None, 10)                1010      
                                                                 
=================================================================
Total params: 105,689,970
Trainable params: 105,689,970
Non-trainable params: 0
_________________________________________________________________
```

In [24]:

```
model_prob1_2.compile(optimizer=keras.optimizers.Nadam(),
                      loss=keras.losses.SparseCategoricalCrossentropy(),
                      metrics=['accuracy'])
```

In [25]:

```
earlystop = keras.callbacks.EarlyStopping(monitor='val_loss',
                                          patience=10)
checkpoint = keras.callbacks.ModelCheckpoint('Model/model_problem1_two.h5',
                                             save_best_only=True)

history = model_prob1_2.fit(X_train_rs, t_train,
                            epochs=100,
                            batch_size=32,
                            validation_data=(X_val_rs, t_val),
                            callbacks=[earlystop, checkpoint])
```

```
Epoch 1/100
42/42 [==============================] - 5s 110ms/step - loss: 122.3389 - accuracy: 0.2474 - val_loss: 9.7673 - val_accuracy: 0.3012
Epoch 2/100
42/42 [==============================] - 5s 109ms/step - loss: 4.6592 - accuracy: 0.4789 - val_loss: 2.4258 - val_accuracy: 0.5422
Epoch 3/100
42/42 [==============================] - 3s 60ms/step - loss: 1.7899 - accuracy: 0.6252 - val_loss: 4.2336 - val_accuracy: 0.3524
Epoch 4/100
42/42 [==============================] - 5s 109ms/step - loss: 1.4020 - accuracy: 0.6983 - val_loss: 2.2027 - val_accuracy: 0.5211
Epoch 5/100
42/42 [==============================] - 4s 108ms/step - loss: 0.5769 - accuracy: 0.8258 - val_loss: 1.9680 - val_accuracy: 0.5723
Epoch 6/100
42/42 [==============================] - 4s 108ms/step - loss: 1.4059 - accuracy: 0.8100 - val_loss: 1.9198 - val_accuracy: 0.5783
Epoch 7/100
42/42 [==============================] - 2s 59ms/step - loss: 0.9238 - accuracy: 0.8258 - val_loss: 3.8463 - val_accuracy: 0.5060
Epoch 8/100
42/42 [==============================] - 2s 59ms/step - loss: 0.3233 - accuracy: 0.9261 - val_loss: 2.6871 - val_accuracy: 0.5753
Epoch 9/100
42/42 [==============================] - 2s 58ms/step - loss: 0.3156 - accuracy: 0.9155 - val_loss: 2.4872 - val_accuracy: 0.5331
Epoch 10/100
42/42 [==============================] - 2s 59ms/step - loss: 0.0867 - accuracy: 0.9766 - val_loss: 2.4670 - val_accuracy: 0.5542
Epoch 11/100
42/42 [==============================] - 2s 59ms/step - loss: 0.1031 - accuracy: 0.9744 - val_loss: 2.6764 - val_accuracy: 0.5241
Epoch 12/100
42/42 [==============================] - 2s 58ms/step - loss: 0.1120 - accuracy: 0.9638 - val_loss: 2.6137 - val_accuracy: 0.5602
Epoch 13/100
42/42 [==============================] - 2s 58ms/step - loss: 0.0833 - accuracy: 0.9751 - val_loss: 2.5957 - val_accuracy: 0.5241
Epoch 14/100
42/42 [==============================] - 2s 59ms/step - loss: 0.0841 - accuracy: 0.9811 - val_loss: 3.9576 - val_accuracy: 0.5241
Epoch 15/100
42/42 [==============================] - 2s 59ms/step - loss: 0.1725 - accuracy: 0.9615 - val_loss: 3.6195 - val_accuracy: 0.5060
Epoch 16/100
42/42 [==============================] - 2s 59ms/step - loss: 0.2085 - accuracy: 0.9668 - val_loss: 4.2052 - val_accuracy: 0.5090
```

In [26]:

```
## Performance result
model = keras.models.load_model('Model/model_problem1_two.h5')
Evaluate_performance(model=model, history=history, Name='CNN')
```

Performance of CNN:

1. In training set:

	precision	recall	f1-score	support
0.0	0.97	0.98	0.97	141
1.0	0.99	1.00	1.00	144
2.0	1.00	0.88	0.94	164
3.0	0.99	0.99	0.99	112
4.0	0.95	0.96	0.96	138
5.0	0.98	1.00	0.99	125
6.0	1.00	0.95	0.97	128
7.0	1.00	0.97	0.99	138
8.0	0.95	1.00	0.97	130
9.0	0.87	1.00	0.93	106
accuracy			0.97	1326
macro avg	0.97	0.97	0.97	1326
weighted avg	0.97	0.97	0.97	1326

Accuracy: 0.9698340874811463
Confusion Matrix

```
[[138   0   0   0   0   1   0   0   1   1]
 [  0 144   0   0   0   0   0   0   0   0]
 [  3   1 144   0   7   0   0   0   0   9]
 [  0   0   0 111   0   0   0   0   1   0]
 [  0   0   0   0 133   0   0   0   1   4]
 [  0   0   0   0   0 125   0   0   0   0]
 [  1   0   0   0   0   2 121   0   2   2]
 [  1   0   0   1   0   0   0 134   2   0]
 [  0   0   0   0   0   0   0   0 130   0]
 [  0   0   0   0   0   0   0   0   0 106]]
```

===

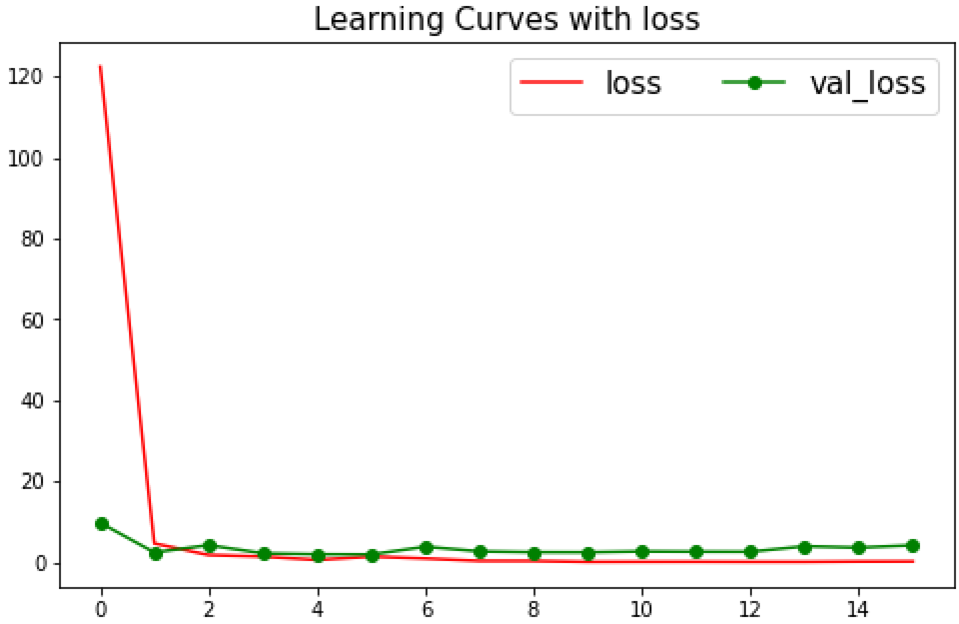
2. In validation set:

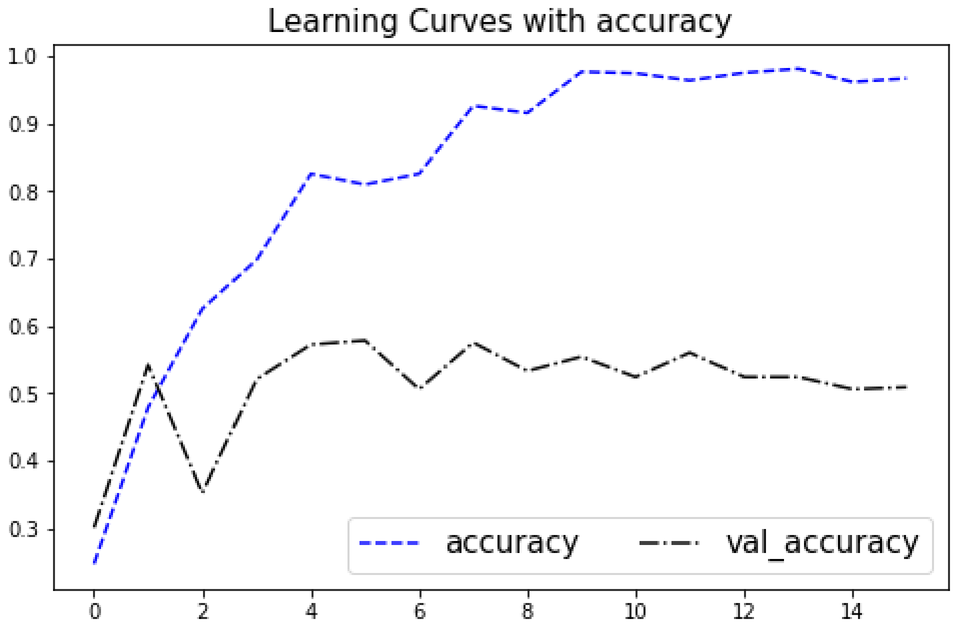
	precision	recall	f1-score	support
0.0	0.61	0.47	0.53	36
1.0	0.64	0.75	0.69	36
2.0	0.64	0.22	0.33	41
3.0	0.63	0.68	0.66	28
4.0	0.63	0.54	0.58	35
5.0	0.61	0.81	0.69	31
6.0	0.79	0.34	0.48	32
7.0	0.82	0.53	0.64	34
8.0	0.55	0.81	0.66	32
9.0	0.33	0.78	0.46	27
accuracy			0.58	332
macro avg	0.63	0.59	0.57	332
weighted avg	0.63	0.58	0.57	332

Accuracy: 0.5783132530120482
Confusion Matrix

```
[[17  2  0  0  2  1  2  0  1 11]
 [ 1 27  0  0  2  1  0  0  2  3]
 [ 1  6  9  1  5  4  0  2  2 11]
 [ 0  1  0 19  1  6  0  0  1  0]
 [ 1  6  1  1 19  1  0  0  2  4]
 [ 0  0  1  2  0 25  0  1  0  2]
 [ 6  0  1  0  1  3 11  0  5  5]
 [ 1  0  2  5  0  0  0 18  4  4]
```

```
[ 0  0  0  2  0  0  0  1 26  3]
[ 1  0  0  0  0  0  1  0  4 21]]
```





In [26]:
```
# Model 3: With transfer learning
base_model = keras.applications.Xception(
    weights='imagenet',
    input_shape=(150, 150, 3),
    include_top=False)

base_model.trainable = False

IMG_SIZE = 150

inputs = keras.Input(shape=(300, 300, 3))
inputs_resized = keras.layers.Resizing(IMG_SIZE, IMG_SIZE)(inputs)
x = base_model(inputs_resized, training=False)
x_pooling = keras.layers.GlobalAveragePooling2D()(x)
outputs = keras.layers.Dense(10, activation='softmax')(x_pooling)
model_prob1_3 = keras.Model(inputs, outputs)
```

In [27]:
```
model_prob1_3.summary()
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_6 (InputLayer)        [(None, 300, 300, 3)]     0

 resizing_2 (Resizing)       (None, 150, 150, 3)       0

 xception (Functional)       (None, 5, 5, 2048)        20861480

 global_average_pooling2d_2   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_9 (Dense)             (None, 10)                20490

=================================================================
Total params: 20,881,970
Trainable params: 20,490
Non-trainable params: 20,861,480
_________________________________________________________________
```

In [28]:
```
model_prob1_3.compile(optimizer=keras.optimizers.Nadam(learning_rate=0.001),
                      loss=keras.losses.SparseCategoricalCrossentropy(),
                      metrics=['accuracy'])
```

In [29]:
```
earlystop = keras.callbacks.EarlyStopping(monitor='val_loss',
                                          patience=10)
checkpoint = keras.callbacks.ModelCheckpoint('Model/model_problem1_three.h5',
                                             save_best_only=True)

history = model_prob1_3.fit(X_train_rs, t_train,
                            epochs=100,
                            batch_size=32,
                            validation_data=(X_val_rs, t_val),
                            callbacks=[earlystop, checkpoint])
```

```
Epoch 1/100
42/42 [==============================] - 3s 38ms/step - loss: 1.5644 - accuracy: 0.5
332 - val_loss: 1.0672 - val_accuracy: 0.6988
Epoch 2/100
42/42 [==============================] - 1s 26ms/step - loss: 0.8523 - accuracy: 0.7
919 - val_loss: 0.8153 - val_accuracy: 0.7229
Epoch 3/100
42/42 [==============================] - 1s 27ms/step - loss: 0.6505 - accuracy: 0.8
318 - val_loss: 0.7074 - val_accuracy: 0.7620
Epoch 4/100
42/42 [==============================] - 1s 30ms/step - loss: 0.5395 - accuracy: 0.8
718 - val_loss: 0.6285 - val_accuracy: 0.8133
Epoch 5/100
42/42 [==============================] - 1s 32ms/step - loss: 0.4689 - accuracy: 0.8
922 - val_loss: 0.5854 - val_accuracy: 0.8012
Epoch 6/100
42/42 [==============================] - 1s 27ms/step - loss: 0.4123 - accuracy: 0.9
080 - val_loss: 0.5474 - val_accuracy: 0.8193
Epoch 7/100
42/42 [==============================] - 1s 28ms/step - loss: 0.3669 - accuracy: 0.9
201 - val_loss: 0.5238 - val_accuracy: 0.8253
Epoch 8/100
42/42 [==============================] - 1s 27ms/step - loss: 0.3324 - accuracy: 0.9
321 - val_loss: 0.5012 - val_accuracy: 0.8283
Epoch 9/100
42/42 [==============================] - 1s 27ms/step - loss: 0.3012 - accuracy: 0.9
359 - val_loss: 0.4826 - val_accuracy: 0.8373
Epoch 10/100
42/42 [==============================] - 1s 27ms/step - loss: 0.2765 - accuracy: 0.9
502 - val_loss: 0.4586 - val_accuracy: 0.8524
Epoch 11/100
42/42 [==============================] - 1s 27ms/step - loss: 0.2531 - accuracy: 0.9
548 - val_loss: 0.4505 - val_accuracy: 0.8645
Epoch 12/100
42/42 [==============================] - 1s 28ms/step - loss: 0.2341 - accuracy: 0.9
600 - val_loss: 0.4379 - val_accuracy: 0.8584
Epoch 13/100
42/42 [==============================] - 1s 27ms/step - loss: 0.2158 - accuracy: 0.9
653 - val_loss: 0.4319 - val_accuracy: 0.8675
Epoch 14/100
42/42 [==============================] - 1s 27ms/step - loss: 0.2010 - accuracy: 0.9
676 - val_loss: 0.4258 - val_accuracy: 0.8494
Epoch 15/100
42/42 [==============================] - 1s 28ms/step - loss: 0.1884 - accuracy: 0.9
713 - val_loss: 0.4225 - val_accuracy: 0.8675
Epoch 16/100
42/42 [==============================] - 1s 27ms/step - loss: 0.1769 - accuracy: 0.9
736 - val_loss: 0.4094 - val_accuracy: 0.8645
Epoch 17/100
42/42 [==============================] - 1s 27ms/step - loss: 0.1665 - accuracy: 0.9
751 - val_loss: 0.3958 - val_accuracy: 0.8765
Epoch 18/100
42/42 [==============================] - 1s 21ms/step - loss: 0.1565 - accuracy: 0.9
789 - val_loss: 0.3973 - val_accuracy: 0.8886
Epoch 19/100
42/42 [==============================] - 1s 27ms/step - loss: 0.1478 - accuracy: 0.9
789 - val_loss: 0.3936 - val_accuracy: 0.8825
Epoch 20/100
42/42 [==============================] - 1s 27ms/step - loss: 0.1392 - accuracy: 0.9
804 - val_loss: 0.3860 - val_accuracy: 0.8765
Epoch 21/100
42/42 [==============================] - 1s 27ms/step - loss: 0.1325 - accuracy: 0.9
834 - val_loss: 0.3770 - val_accuracy: 0.8735
Epoch 22/100
```

```
42/42 [==============================] - 1s 27ms/step - loss: 0.1245 - accuracy: 0.9
827 - val_loss: 0.3758 - val_accuracy: 0.8886
Epoch 23/100
42/42 [==============================] - 1s 21ms/step - loss: 0.1181 - accuracy: 0.9
879 - val_loss: 0.3825 - val_accuracy: 0.8705
Epoch 24/100
42/42 [==============================] - 1s 28ms/step - loss: 0.1128 - accuracy: 0.9
857 - val_loss: 0.3709 - val_accuracy: 0.8825
Epoch 25/100
42/42 [==============================] - 1s 27ms/step - loss: 0.1071 - accuracy: 0.9
857 - val_loss: 0.3676 - val_accuracy: 0.8825
Epoch 26/100
42/42 [==============================] - 1s 27ms/step - loss: 0.1019 - accuracy: 0.9
887 - val_loss: 0.3610 - val_accuracy: 0.8886
Epoch 27/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0967 - accuracy: 0.9
887 - val_loss: 0.3611 - val_accuracy: 0.8855
Epoch 28/100
42/42 [==============================] - 1s 27ms/step - loss: 0.0928 - accuracy: 0.9
902 - val_loss: 0.3593 - val_accuracy: 0.8855
Epoch 29/100
42/42 [==============================] - 1s 28ms/step - loss: 0.0892 - accuracy: 0.9
894 - val_loss: 0.3515 - val_accuracy: 0.8916
Epoch 30/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0847 - accuracy: 0.9
925 - val_loss: 0.3555 - val_accuracy: 0.8765
Epoch 31/100
42/42 [==============================] - 1s 29ms/step - loss: 0.0815 - accuracy: 0.9
932 - val_loss: 0.3483 - val_accuracy: 0.8855
Epoch 32/100
42/42 [==============================] - 1s 29ms/step - loss: 0.0776 - accuracy: 0.9
940 - val_loss: 0.3458 - val_accuracy: 0.8855
Epoch 33/100
42/42 [==============================] - 1s 22ms/step - loss: 0.0746 - accuracy: 0.9
955 - val_loss: 0.3480 - val_accuracy: 0.8886
Epoch 34/100
42/42 [==============================] - 1s 28ms/step - loss: 0.0715 - accuracy: 0.9
955 - val_loss: 0.3447 - val_accuracy: 0.8886
Epoch 35/100
42/42 [==============================] - 1s 22ms/step - loss: 0.0686 - accuracy: 0.9
955 - val_loss: 0.3448 - val_accuracy: 0.8886
Epoch 36/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0659 - accuracy: 0.9
970 - val_loss: 0.3462 - val_accuracy: 0.8886
Epoch 37/100
42/42 [==============================] - 1s 27ms/step - loss: 0.0633 - accuracy: 0.9
962 - val_loss: 0.3429 - val_accuracy: 0.8855
Epoch 38/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0612 - accuracy: 0.9
977 - val_loss: 0.3434 - val_accuracy: 0.8795
Epoch 39/100
42/42 [==============================] - 1s 22ms/step - loss: 0.0589 - accuracy: 0.9
970 - val_loss: 0.3431 - val_accuracy: 0.8795
Epoch 40/100
42/42 [==============================] - 1s 29ms/step - loss: 0.0567 - accuracy: 0.9
970 - val_loss: 0.3402 - val_accuracy: 0.8825
Epoch 41/100
42/42 [==============================] - 1s 29ms/step - loss: 0.0543 - accuracy: 0.9
985 - val_loss: 0.3384 - val_accuracy: 0.8886
Epoch 42/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0524 - accuracy: 0.9
985 - val_loss: 0.3415 - val_accuracy: 0.8886
Epoch 43/100
42/42 [==============================] - 1s 28ms/step - loss: 0.0507 - accuracy: 0.9
```

```
992 - val_loss: 0.3383 - val_accuracy: 0.8855
Epoch 44/100
42/42 [==============================] - 1s 28ms/step - loss: 0.0490 - accuracy: 0.9
985 - val_loss: 0.3376 - val_accuracy: 0.8825
Epoch 45/100
42/42 [==============================] - 1s 27ms/step - loss: 0.0473 - accuracy: 0.9
992 - val_loss: 0.3352 - val_accuracy: 0.8855
Epoch 46/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0456 - accuracy: 0.9
992 - val_loss: 0.3360 - val_accuracy: 0.8825
Epoch 47/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0441 - accuracy: 0.9
992 - val_loss: 0.3377 - val_accuracy: 0.8916
Epoch 48/100
42/42 [==============================] - 1s 32ms/step - loss: 0.0428 - accuracy: 0.9
992 - val_loss: 0.3347 - val_accuracy: 0.8886
Epoch 49/100
42/42 [==============================] - 1s 27ms/step - loss: 0.0414 - accuracy: 0.9
992 - val_loss: 0.3341 - val_accuracy: 0.8825
Epoch 50/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0400 - accuracy: 0.9
992 - val_loss: 0.3345 - val_accuracy: 0.8916
Epoch 51/100
42/42 [==============================] - 1s 22ms/step - loss: 0.0388 - accuracy: 0.9
992 - val_loss: 0.3347 - val_accuracy: 0.8886
Epoch 52/100
42/42 [==============================] - 1s 28ms/step - loss: 0.0373 - accuracy: 0.9
992 - val_loss: 0.3323 - val_accuracy: 0.8886
Epoch 53/100
42/42 [==============================] - 1s 20ms/step - loss: 0.0362 - accuracy: 0.9
992 - val_loss: 0.3336 - val_accuracy: 0.8886
Epoch 54/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0352 - accuracy: 0.9
992 - val_loss: 0.3356 - val_accuracy: 0.8855
Epoch 55/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0341 - accuracy: 0.9
992 - val_loss: 0.3332 - val_accuracy: 0.8886
Epoch 56/100
42/42 [==============================] - 1s 27ms/step - loss: 0.0331 - accuracy: 0.9
992 - val_loss: 0.3317 - val_accuracy: 0.8916
Epoch 57/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0321 - accuracy: 0.9
992 - val_loss: 0.3341 - val_accuracy: 0.8916
Epoch 58/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0311 - accuracy: 0.9
992 - val_loss: 0.3347 - val_accuracy: 0.8916
Epoch 59/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0301 - accuracy: 1.0
000 - val_loss: 0.3364 - val_accuracy: 0.8916
Epoch 60/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0295 - accuracy: 1.0
000 - val_loss: 0.3325 - val_accuracy: 0.8855
Epoch 61/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0285 - accuracy: 1.0
000 - val_loss: 0.3342 - val_accuracy: 0.8916
Epoch 62/100
42/42 [==============================] - 1s 20ms/step - loss: 0.0277 - accuracy: 1.0
000 - val_loss: 0.3360 - val_accuracy: 0.8916
Epoch 63/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0271 - accuracy: 1.0
000 - val_loss: 0.3327 - val_accuracy: 0.8886
Epoch 64/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0261 - accuracy: 1.0
000 - val_loss: 0.3347 - val_accuracy: 0.8886
```

```
Epoch 65/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0255 - accuracy: 1.0
000 - val_loss: 0.3345 - val_accuracy: 0.8886
Epoch 66/100
42/42 [==============================] - 1s 21ms/step - loss: 0.0247 - accuracy: 1.0
000 - val_loss: 0.3348 - val_accuracy: 0.8916
```

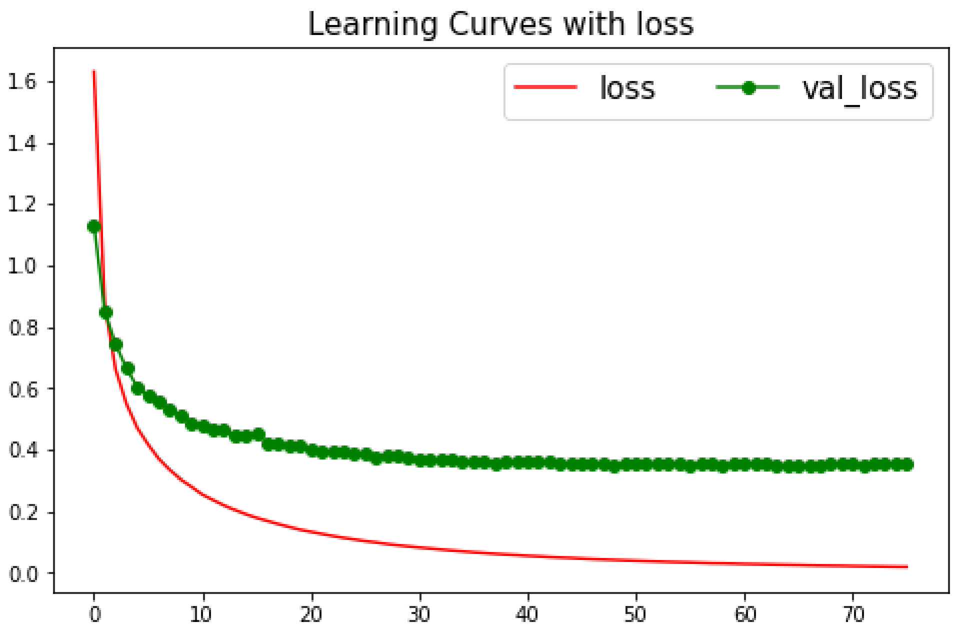
In [18]:

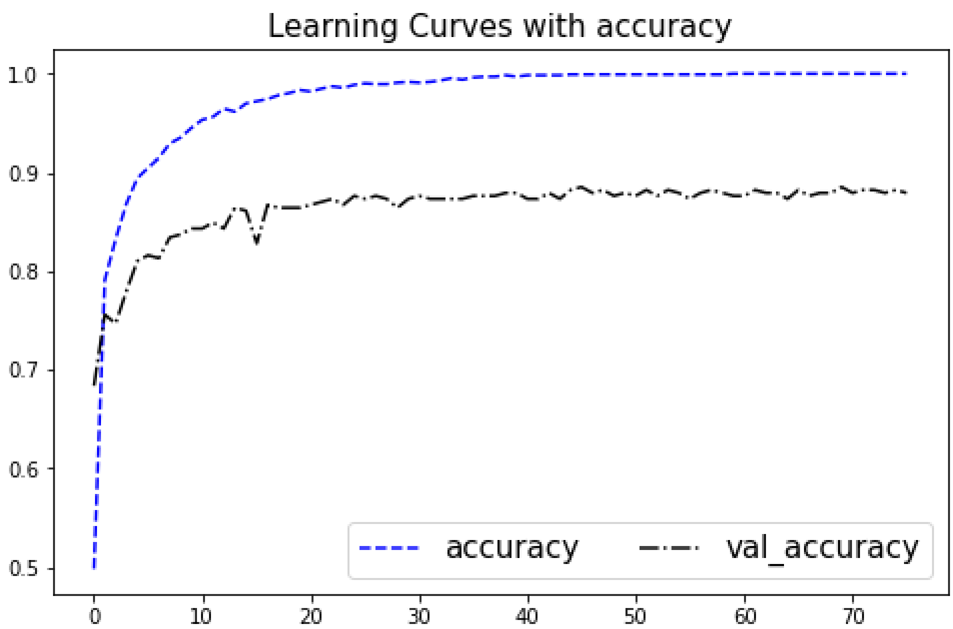
```
## Performance result
model = keras.models.load_model('Model/model_problem1_three.h5')
Evaluate_performance(model=model, history=history, Name='Transfer learning')
```

Performance of Transfer learning:

```
1. In training set:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       141
         1.0       1.00      1.00      1.00       144
         2.0       1.00      1.00      1.00       164
         3.0       1.00      1.00      1.00       112
         4.0       1.00      1.00      1.00       138
         5.0       1.00      1.00      1.00       125
         6.0       1.00      1.00      1.00       128
         7.0       1.00      1.00      1.00       138
         8.0       1.00      1.00      1.00       130
         9.0       1.00      1.00      1.00       106

    accuracy                           1.00      1326
   macro avg       1.00      1.00      1.00      1326
weighted avg       1.00      1.00      1.00      1326

Accuracy: 1.0
Confusion Matrix
[[141   0   0   0   0   0   0   0   0   0]
 [  0 144   0   0   0   0   0   0   0   0]
 [  0   0 164   0   0   0   0   0   0   0]
 [  0   0   0 112   0   0   0   0   0   0]
 [  0   0   0   0 138   0   0   0   0   0]
 [  0   0   0   0   0 125   0   0   0   0]
 [  0   0   0   0   0   0 128   0   0   0]
 [  0   0   0   0   0   0   0 138   0   0]
 [  0   0   0   0   0   0   0   0 130   0]
 [  0   0   0   0   0   0   0   0   0 106]]

 =====================================================

2. In validation set:
              precision    recall  f1-score   support

         0.0       0.78      0.81      0.79        36
         1.0       0.89      0.94      0.92        36
         2.0       0.84      0.78      0.81        41
         3.0       0.96      0.93      0.95        28
         4.0       0.94      0.91      0.93        35
         5.0       0.97      0.97      0.97        31
         6.0       0.84      0.84      0.84        32
         7.0       0.82      0.94      0.88        34
         8.0       0.94      0.91      0.92        32
         9.0       0.88      0.81      0.85        27

    accuracy                           0.88       332
   macro avg       0.89      0.88      0.89       332
weighted avg       0.88      0.88      0.88       332

Accuracy: 0.8825301204819277
Confusion Matrix
[[29  1  0  0  0  1  2  0  0  3]
 [ 0 34  1  0  0  0  1  0  0  0]
 [ 1  2 32  0  1  0  1  4  0  0]
 [ 0  0  0 26  0  0  1  1  0  0]
 [ 0  1  2  0 32  0  0  0  0  0]
 [ 0  0  0  0  1 30  0  0  0  0]
 [ 2  0  1  1  0  0 27  1  0  0]
 [ 1  0  1  0  0  0  0 32  0  0]
```

```
[ 2  0  0  0  0  0  0  1 29  0]
[ 2  0  1  0  0  0  0  0  2 22]]
```





Problem 3

In [2]:

```
bbox = pd.read_csv('car_detection_dataset/train_bounding_boxes.csv')

N = len(bbox)

# Create a numpy array with all images
for i in range(N):
    filename='car_detection_dataset/training_images/'+bbox['image'][i]
    image = np.array(Image.open(filename))
    image_col = image.ravel()[:,np.newaxis]

    if i==0:
        X_train_full = image_col
    else:
        X_train_full = np.hstack((X_train_full, image_col))

# Training feature matrices
X_train_full = X_train_full.T
```

```
# Training labels
t_train_full = bbox.drop('image', axis=1).round().to_numpy().astype(int)

X_train_full.shape, t_train_full.shape
```

Out[2]: ((559, 770640), (559, 4))

In [3]:
```
# size of each RGB image
(Nx,Ny,Nz) = image.shape

Nx, Ny, Nz
```

Out[3]: (380, 676, 3)

In [4]:
```
# Scale and split the training data and the target
X_train_scaled = X_train_full / 255.0
t_train_scaled = np.vstack((t_train_full[:,0]/Ny, t_train_full[:,1]/Nx, t_train_ful

X_val, X_train = X_train_scaled[:50], X_train_scaled[50:]
t_val, t_train = t_train_scaled[:50], t_train_scaled[50:]

X_train.shape, t_train.shape, X_val.shape, t_val.shape
```

Out[4]: ((509, 770640), (509, 4), (50, 770640), (50, 4))

In [15]:
```
# Reshape the data
X_train_rs = tf.constant(X_train.reshape((X_train.shape[0],Nx,Ny,3)),
                         dtype=tf.float16)
X_val_rs = tf.constant(X_val.reshape((X_val.shape[0], Nx,Ny,3)),
                       dtype=tf.float16)

X_train_rs.shape, X_val_rs.shape
```

Out[15]: (TensorShape([509, 380, 676, 3]), TensorShape([50, 380, 676, 3]))

In [20]:
```
# Build the model
base_model = keras.applications.VGG16(
    weights='imagenet',
    input_tensor=keras.layers.Input(shape=(224, 224, 3)),
    include_top=False)

base_model.trainable = False

inputs = keras.Input(shape=(Nx, Ny, Nz))
inputs_resized = keras.layers.Resizing(224, 224)(inputs)
x = base_model(inputs_resized, training=False)
x_flatten = keras.layers.Flatten()(x)
layer1 = keras.layers.Dense(128, activation="selu", kernel_initializer='lecun_norma
layer2 = keras.layers.Dense(64, activation="selu", kernel_initializer='lecun_normal'
layer3 = keras.layers.Dense(32, activation="selu", kernel_initializer='lecun_normal'
outputs = keras.layers.Dense(4, activation="sigmoid")(layer3)
model_prob2 = keras.Model(inputs, outputs)
```

In [21]:
```
model_prob2.summary()
```

```
Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_10 (InputLayer)       [(None, 380, 676, 3)]     0

 resizing_4 (Resizing)       (None, 224, 224, 3)       0

 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten_4 (Flatten)         (None, 25088)             0

 dense_16 (Dense)            (None, 128)               3211392

 dense_17 (Dense)            (None, 64)                8256

 dense_18 (Dense)            (None, 32)                2080

 dense_19 (Dense)            (None, 4)                 132

=================================================================
Total params: 17,936,548
Trainable params: 3,221,860
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [22]:
```python
model_prob2.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001),
                    loss=keras.losses.MeanSquaredError())
```

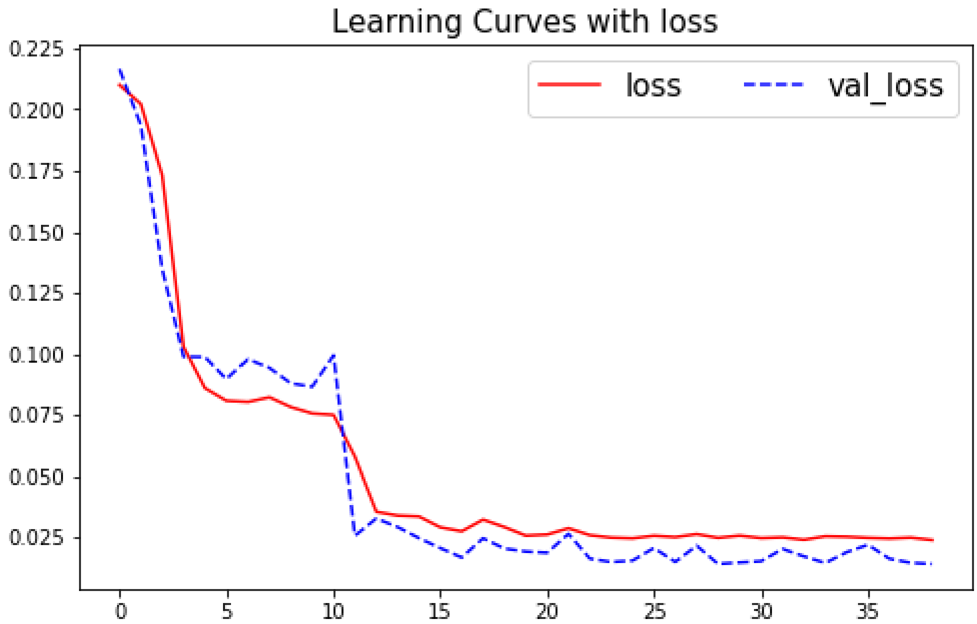
In [23]:
```python
earlystop = keras.callbacks.EarlyStopping(monitor='val_loss',
                                          patience=10)
checkpoint = keras.callbacks.ModelCheckpoint('Model/model_problem2.h5',
                                             save_best_only=True)

history = model_prob2.fit(X_train_rs, t_train,
                          epochs=50,
                          batch_size=10,
                          validation_data=(X_val_rs, t_val),
                          callbacks=[earlystop, checkpoint])
```

```
Epoch 1/50
51/51 [==============================] - 1s 17ms/step - loss: 0.2100 - val_loss: 0.2
165
Epoch 2/50
51/51 [==============================] - 1s 15ms/step - loss: 0.2022 - val_loss: 0.1
936
Epoch 3/50
51/51 [==============================] - 1s 14ms/step - loss: 0.1733 - val_loss: 0.1
348
Epoch 4/50
51/51 [==============================] - 1s 14ms/step - loss: 0.1028 - val_loss: 0.0
988
Epoch 5/50
51/51 [==============================] - 1s 15ms/step - loss: 0.0860 - val_loss: 0.0
987
Epoch 6/50
51/51 [==============================] - 1s 14ms/step - loss: 0.0808 - val_loss: 0.0
896
Epoch 7/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0803 - val_loss: 0.0
978
Epoch 8/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0822 - val_loss: 0.0
942
Epoch 9/50
51/51 [==============================] - 1s 14ms/step - loss: 0.0782 - val_loss: 0.0
879
Epoch 10/50
51/51 [==============================] - 1s 14ms/step - loss: 0.0756 - val_loss: 0.0
865
Epoch 11/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0750 - val_loss: 0.0
994
Epoch 12/50
51/51 [==============================] - 1s 14ms/step - loss: 0.0579 - val_loss: 0.0
256
Epoch 13/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0355 - val_loss: 0.0
327
Epoch 14/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0339 - val_loss: 0.0
292
Epoch 15/50
51/51 [==============================] - 1s 15ms/step - loss: 0.0335 - val_loss: 0.0
247
Epoch 16/50
51/51 [==============================] - 1s 15ms/step - loss: 0.0291 - val_loss: 0.0
206
Epoch 17/50
51/51 [==============================] - 1s 15ms/step - loss: 0.0275 - val_loss: 0.0
167
Epoch 18/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0323 - val_loss: 0.0
246
Epoch 19/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0292 - val_loss: 0.0
204
Epoch 20/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0257 - val_loss: 0.0
192
Epoch 21/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0261 - val_loss: 0.0
187
Epoch 22/50
```

```
51/51 [==============================] - 1s 11ms/step - loss: 0.0287 - val_loss: 0.0
264
Epoch 23/50
51/51 [==============================] - 1s 14ms/step - loss: 0.0259 - val_loss: 0.0
163
Epoch 24/50
51/51 [==============================] - 1s 14ms/step - loss: 0.0249 - val_loss: 0.0
149
Epoch 25/50
51/51 [==============================] - 1s 12ms/step - loss: 0.0246 - val_loss: 0.0
155
Epoch 26/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0256 - val_loss: 0.0
206
Epoch 27/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0251 - val_loss: 0.0
149
Epoch 28/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0263 - val_loss: 0.0
216
Epoch 29/50
51/51 [==============================] - 1s 14ms/step - loss: 0.0248 - val_loss: 0.0
141
Epoch 30/50
51/51 [==============================] - 1s 12ms/step - loss: 0.0257 - val_loss: 0.0
147
Epoch 31/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0247 - val_loss: 0.0
152
Epoch 32/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0250 - val_loss: 0.0
202
Epoch 33/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0240 - val_loss: 0.0
172
Epoch 34/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0254 - val_loss: 0.0
145
Epoch 35/50
51/51 [==============================] - 1s 12ms/step - loss: 0.0252 - val_loss: 0.0
189
Epoch 36/50
51/51 [==============================] - 1s 12ms/step - loss: 0.0248 - val_loss: 0.0
220
Epoch 37/50
51/51 [==============================] - 1s 11ms/step - loss: 0.0245 - val_loss: 0.0
162
Epoch 38/50
51/51 [==============================] - 1s 12ms/step - loss: 0.0249 - val_loss: 0.0
145
Epoch 39/50
51/51 [==============================] - 1s 12ms/step - loss: 0.0239 - val_loss: 0.0
141
```

In [24]:

```python
key_names = list(history.history.keys())
colors = ['-r','--b']

plt.figure(figsize=(8,5))
for i in [0,1]:
    plt.plot(history.history[key_names[i]], colors[i], label=key_names[i])
plt.legend(fontsize=15,ncol=2)
plt.title('Learning Curves with loss', size=15);
```


Learning Curves with loss
loss
val_loss
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0
5
10
15
20
25
30
35

```
In [ ]:
```