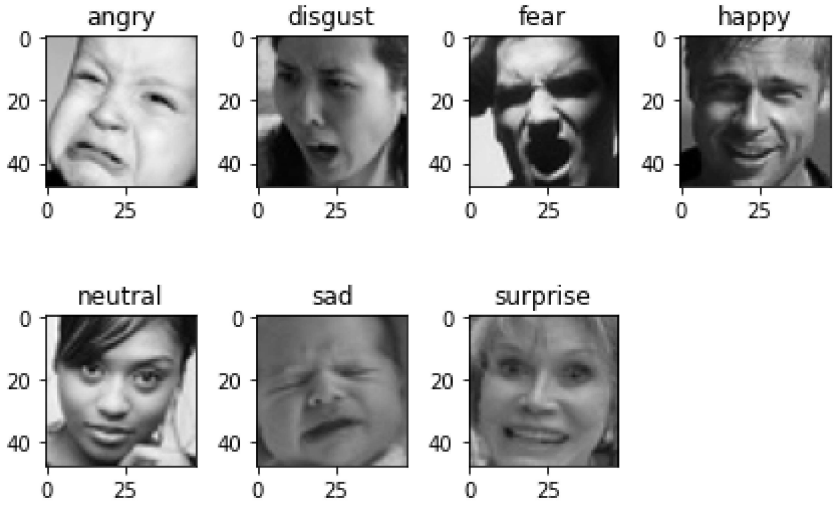
In [1]:
```
# Importing Libraries
import os

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

from keras.preprocessing.image import load_img, img_to_array           # for working with Images
from keras.preprocessing.image import ImageDataGenerator               # data Augumentation
from keras.callbacks import EarlyStopping                              # for finding optimal model

from keras.layers import Dense, Dropout, Flatten, Conv2D, BatchNormalization, Activation, MaxPooling2D
from keras.models import Model, Sequential

from tensorflow.keras.applications import MobileNet                   # transfer learning

from sklearn.metrics import classification_report                      # for result
```

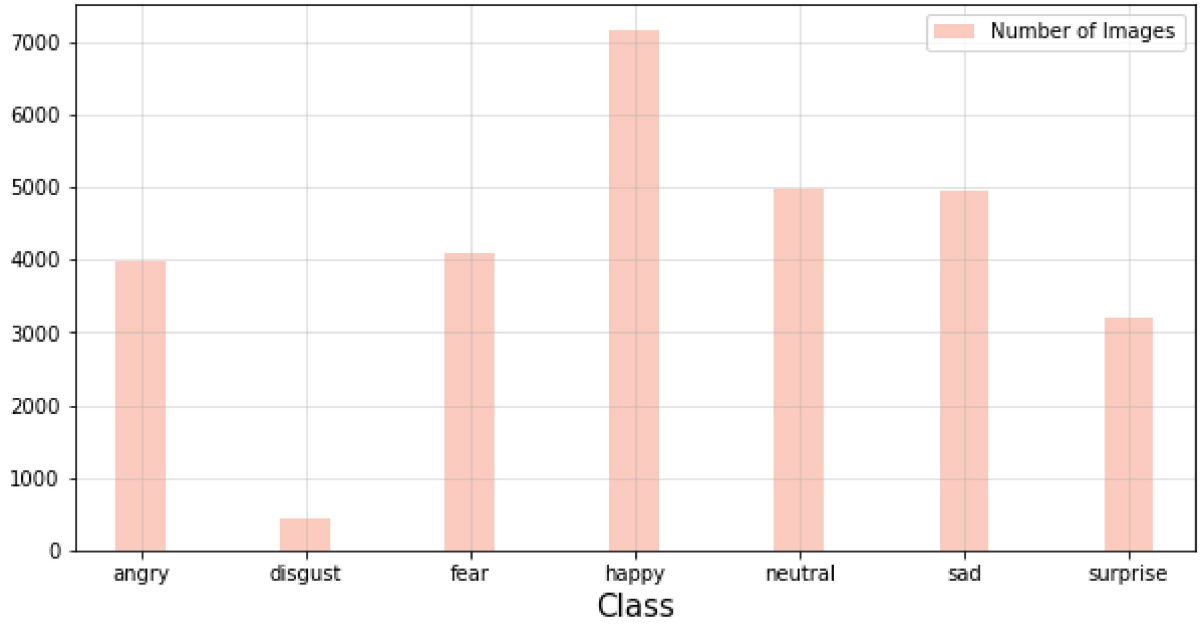
Intial Steps

In [2]:
```
# input path for the images

train_folder_path = r'data\train'
test_folder_path = r'data\test'
```

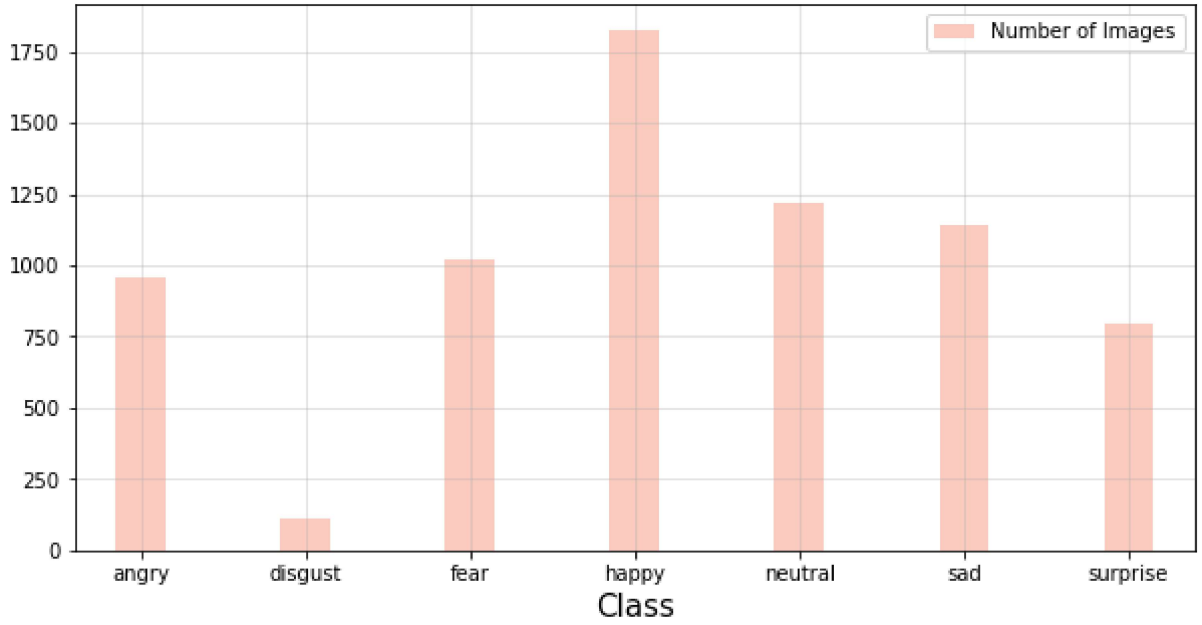
In [3]:
```
# size of the image: 48*48 pixels
image_size = 48

# for plotting subplots
plot_no = 1

# image to view
random_image_no = 4

for each_folder in os.listdir(train_folder_path):

    complete_folder_path = os.path.join(train_folder_path , each_folder)
    first_image_path = os.path.join( complete_folder_path , os.listdir(complete_folder_path)[random_image_no] )

    image = load_img( first_image_path , target_size = (image_size, image_size))

    plt.subplot(2,4,plot_no)
    plt.imshow( image  , cmap = "gray")
    plt.title(each_folder)

    plot_no += 1

plt.tight_layout()
plt.show()
```



In [4]:
```
# traing images info
data_info = pd.DataFrame({'Class' : None , 'Number of Images' : None} , index = range(6))

c = 0
for each_folder in os.listdir(train_folder_path):
    data_info.at[c , 'Class'] = each_folder
    data_info.at[c , 'Number of Images'] = len(os.listdir(os.path.join(train_folder_path , each_folder)))
    c += 1

data_info.set_index('Class' , inplace = True)

data_info.plot(kind = 'bar' , color = '#FCCBC0' , figsize = (10, 5) , rot = 0 , width = 0.3)
plt.grid('True' , alpha = 0.4)
plt.xlabel('Class' , size = 15)
plt.show()
```



In [5]:
```
# testing images info
data_info = pd.DataFrame({'Class' : None , 'Number of Images' : None} , index = range(6))

c = 0
for each_folder in os.listdir(test_folder_path):
    data_info.at[c , 'Class'] = each_folder
    data_info.at[c , 'Number of Images'] = len(os.listdir(os.path.join(test_folder_path , each_folder)))
    c += 1

data_info.set_index('Class' , inplace = True)

data_info.plot(kind = 'bar' , color = '#FCCBC0' , figsize = (10, 5) , rot = 0 , width = 0.3)
plt.grid('True' , alpha = 0.4)
plt.xlabel('Class' , size = 15)
plt.show()
```



In [6]:
```
# building train data generator

batch_size = 256  # determining number of samples to be taken

train_datagen = ImageDataGenerator(rescale = 1.0/255.0,                    # data augumentation
                                   width_shift_range = 0.1,
                                   height_shift_range = 0.1,
                                   rotation_range = 20,
                                   horizontal_flip = True)


train_generator = train_datagen.flow_from_directory(train_folder_path,          # processing images
                                                    target_size = (48,48),
                                                    color_mode = "grayscale",
                                                    batch_size = batch_size,
                                                    class_mode = 'categorical',
                                                    shuffle = True)
```

```
Found 28820 images belonging to 7 classes.
```

In [7]:
```
# building test data generator
test_datagen = ImageDataGenerator(rescale = 1.0/255)

test_generator = test_datagen.flow_from_directory(test_folder_path,          # processing images
                                                  target_size = (48,48),
                                                  color_mode = "grayscale",
                                                  batch_size = batch_size,
                                                  class_mode = 'categorical',
                                                  shuffle = False)
```

```
Found 7066 images belonging to 7 classes.
```

Model 1

In [8]:

```
# Initialising the model
model = Sequential()

# 1 - Convolution
model.add( Conv2D( 64 , ( 3 , 3 ) , input_shape = (48, 48 , 1 ) ) )
model.add( BatchNormalization() )
model.add( Activation('relu') )
model.add( MaxPooling2D(pool_size = (2, 2) ) )
model.add( Dropout(0.25) )

# 2nd Convolution layer
model.add(Conv2D(128,(5,5)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

# 3rd Convolution layer
model.add(Conv2D(512,(3,3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

# Flattening
model.add(Flatten())

# Fully connected layer 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(7, activation = 'softmax'))  # 7 --> total number of classes

print(model.summary())

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

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 46, 64)	640
batch_normalization (BatchNo	(None, 46, 46, 64)	256
activation (Activation)	(None, 46, 46, 64)	0
max_pooling2d (MaxPooling2D)	(None, 23, 23, 64)	0
dropout (Dropout)	(None, 23, 23, 64)	0
conv2d_1 (Conv2D)	(None, 19, 19, 128)	204928
batch_normalization_1 (Batch	(None, 19, 19, 128)	512
activation_1 (Activation)	(None, 19, 19, 128)	0
max_pooling2d_1 (MaxPooling2	(None, 9, 9, 128)	0
dropout_1 (Dropout)	(None, 9, 9, 128)	0
conv2d_2 (Conv2D)	(None, 7, 7, 512)	590336
batch_normalization_2 (Batch	(None, 7, 7, 512)	2048
activation_2 (Activation)	(None, 7, 7, 512)	0
max_pooling2d_2 (MaxPooling2	(None, 3, 3, 512)	0
dropout_2 (Dropout)	(None, 3, 3, 512)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
batch_normalization_3 (Batch	(None, 256)	1024
activation_3 (Activation)	(None, 256)	0

```
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 1799
=================================================================
Total params: 1,981,447
Trainable params: 1,979,527
Non-trainable params: 1,920
_________________________________________________________________
None
```

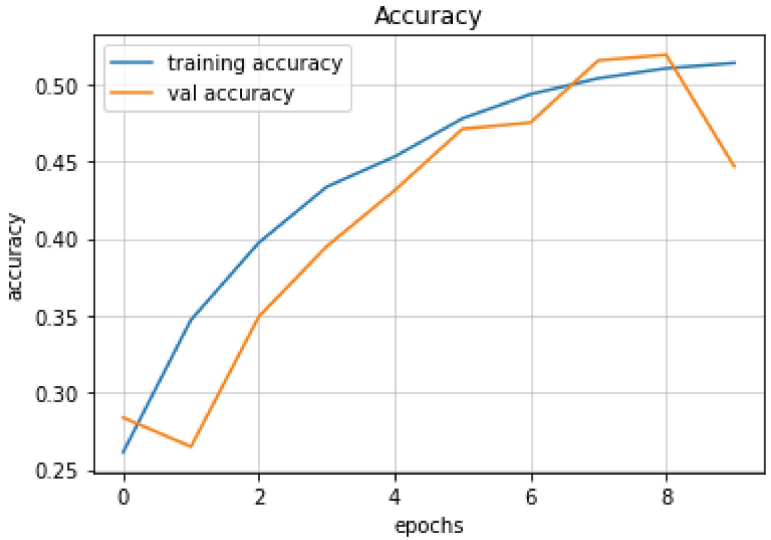
In [9]:

```
# Early stopping
es = EarlyStopping( monitor ='val_loss', min_delta = 0.05 ,
                    patience = 2 ,  mode = 'min' , restore_best_weights = True)
```

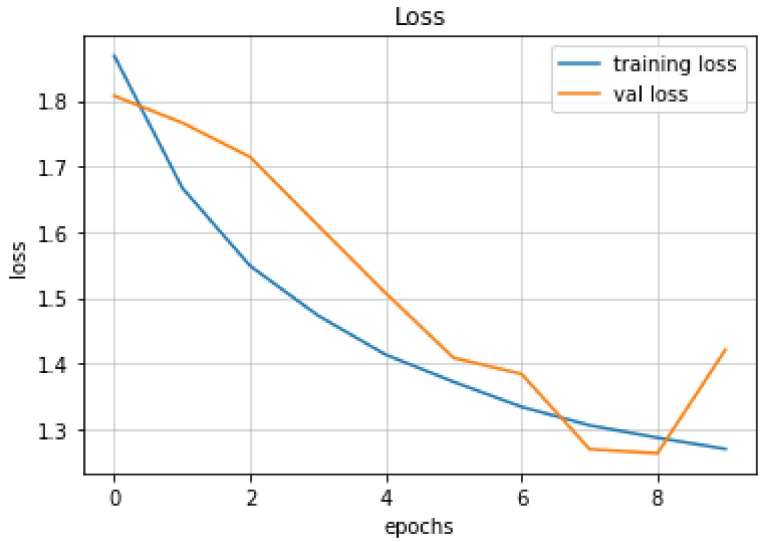
In [10]:
```
# number of epochs to train the NN
epochs = 10

history = model.fit(train_generator,
                    steps_per_epoch = train_generator.n//train_generator.batch_size,
                    epochs = epochs,
                    validation_data = test_generator,
                    validation_steps = test_generator.n//test_generator.batch_size ,
                    callbacks = [es,]
                   )
```

```
Epoch 1/10
112/112 [==============================] - 195s 2s/step - loss: 1.8685 - accuracy: 0.2613 - val_loss: 1.8075 - val_ac
curacy: 0.2839
Epoch 2/10
112/112 [==============================] - 191s 2s/step - loss: 1.6677 - accuracy: 0.3473 - val_loss: 1.7668 - val_ac
curacy: 0.2650
Epoch 3/10
112/112 [==============================] - 188s 2s/step - loss: 1.5489 - accuracy: 0.3974 - val_loss: 1.7145 - val_ac
curacy: 0.3494
Epoch 4/10
112/112 [==============================] - 185s 2s/step - loss: 1.4733 - accuracy: 0.4337 - val_loss: 1.6105 - val_ac
curacy: 0.3950
Epoch 5/10
112/112 [==============================] - 184s 2s/step - loss: 1.4136 - accuracy: 0.4532 - val_loss: 1.5071 - val_ac
curacy: 0.4313
Epoch 6/10
112/112 [==============================] - 183s 2s/step - loss: 1.3723 - accuracy: 0.4780 - val_loss: 1.4087 - val_ac
curacy: 0.4712
Epoch 7/10
112/112 [==============================] - 183s 2s/step - loss: 1.3343 - accuracy: 0.4937 - val_loss: 1.3843 - val_ac
curacy: 0.4753
Epoch 8/10
112/112 [==============================] - 183s 2s/step - loss: 1.3062 - accuracy: 0.5040 - val_loss: 1.2699 - val_ac
curacy: 0.5156
Epoch 9/10
112/112 [==============================] - 183s 2s/step - loss: 1.2873 - accuracy: 0.5105 - val_loss: 1.2636 - val_ac
curacy: 0.5192
Epoch 10/10
112/112 [==============================] - 183s 2s/step - loss: 1.2703 - accuracy: 0.5139 - val_loss: 1.4213 - val_ac
curacy: 0.4469
```

In [11]:
```
# plotting graphs for accuracy

plt.plot(history.history['accuracy'], label = 'training accuracy')
plt.plot(history.history['val_accuracy'], label = 'val accuracy')

plt.xlabel('epochs')
plt.ylabel('accuracy')

plt.title('Accuracy')

plt.grid(alpha = 0.6)
plt.legend()
plt.show()
```



In [12]:
```
# plotting for loss

plt.plot(history.history['loss'], label = 'training loss')
plt.plot(history.history['val_loss'], label = 'val loss')

plt.title('Loss')

plt.xlabel('epochs')
plt.ylabel('loss')

plt.grid(alpha = 0.6)
plt.legend()
plt.show()
```



In [13]:
```
pred = model.predict(test_generator)

pred_labels = np.argmax(pred, axis = -1)
```

In [14]:
```
print(classification_report(test_generator.classes , pred_labels , zero_division = 0))
```

```
              precision    recall  f1-score   support

           0       0.48      0.37      0.42       960
           1       0.68      0.12      0.20       111
           2       0.30      0.32      0.31      1018
           3       0.85      0.67      0.75      1825
           4       0.54      0.40      0.46      1216
           5       0.37      0.59      0.46      1139
           6       0.57      0.76      0.65       797

    accuracy                           0.52      7066
   macro avg       0.54      0.46      0.46      7066
weighted avg       0.56      0.52      0.52      7066
```

Model 2

In [15]:

```
# Initialising the model
model = Sequential()

# 1 - Convolution
model.add( Conv2D( 64 , ( 3 , 3 ) , input_shape = (48, 48 , 1 ) ) )
model.add( BatchNormalization() )
model.add( Activation('relu') )
model.add( MaxPooling2D(pool_size = (2, 2) ) )
model.add( Dropout(0.2) )

# 2nd Convolution layer
model.add(Conv2D(128,(3,3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))

# 3rd Convolution layer
model.add(Conv2D(512,(3,3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))

# 4th Convolution layer
model.add(Conv2D(512,(3,3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))

# Flattening
model.add(Flatten())

# Fully connected layer 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.2))

# Fully connected layer 2nd layer
```

```
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.2))

model.add(Dense(7, activation = 'softmax'))  # 7 --> total number of classes

print(model.summary())

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

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 46, 46, 64)	640
batch_normalization_4 (Batch	(None, 46, 46, 64)	256
activation_4 (Activation)	(None, 46, 46, 64)	0
max_pooling2d_3 (MaxPooling2	(None, 23, 23, 64)	0
dropout_4 (Dropout)	(None, 23, 23, 64)	0
conv2d_4 (Conv2D)	(None, 21, 21, 128)	73856
batch_normalization_5 (Batch	(None, 21, 21, 128)	512
activation_5 (Activation)	(None, 21, 21, 128)	0
max_pooling2d_4 (MaxPooling2	(None, 10, 10, 128)	0
dropout_5 (Dropout)	(None, 10, 10, 128)	0
conv2d_5 (Conv2D)	(None, 8, 8, 512)	590336
batch_normalization_6 (Batch	(None, 8, 8, 512)	2048
activation_6 (Activation)	(None, 8, 8, 512)	0
max_pooling2d_5 (MaxPooling2	(None, 4, 4, 512)	0
dropout_6 (Dropout)	(None, 4, 4, 512)	0
conv2d_6 (Conv2D)	(None, 2, 2, 512)	2359808
batch_normalization_7 (Batch	(None, 2, 2, 512)	2048
activation_7 (Activation)	(None, 2, 2, 512)	0
max_pooling2d_6 (MaxPooling2	(None, 1, 1, 512)	0

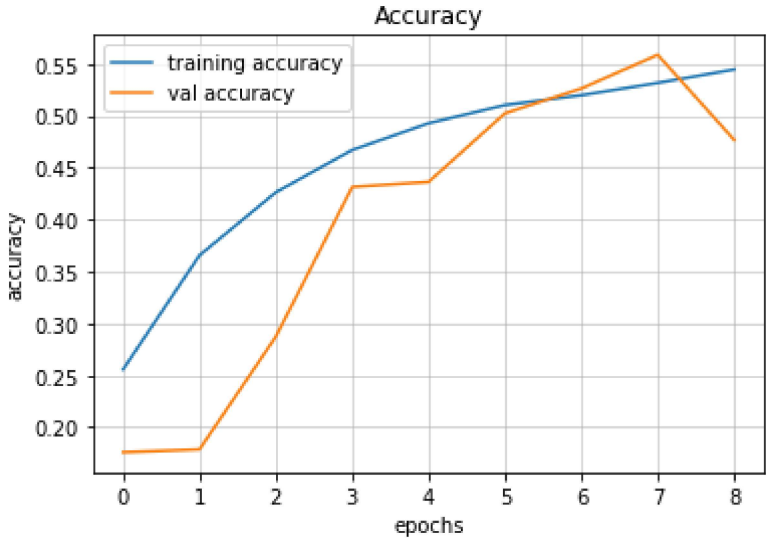
```
dropout_7 (Dropout)          (None, 1, 1, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
batch_normalization_8 (Batch (None, 256)               1024
_________________________________________________________________
activation_8 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_8 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_9 (Batch (None, 512)               2048
_________________________________________________________________
activation_9 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_9 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 7)                 3591
=================================================================
Total params: 3,299,079
Trainable params: 3,295,111
Non-trainable params: 3,968
_________________________________________________________________
None
```

In [16]:
```
# Early stopping
es = EarlyStopping( monitor ='val_loss', min_delta = 0.05 ,
                    patience = 2 ,  mode = 'min' , restore_best_weights = True)
```

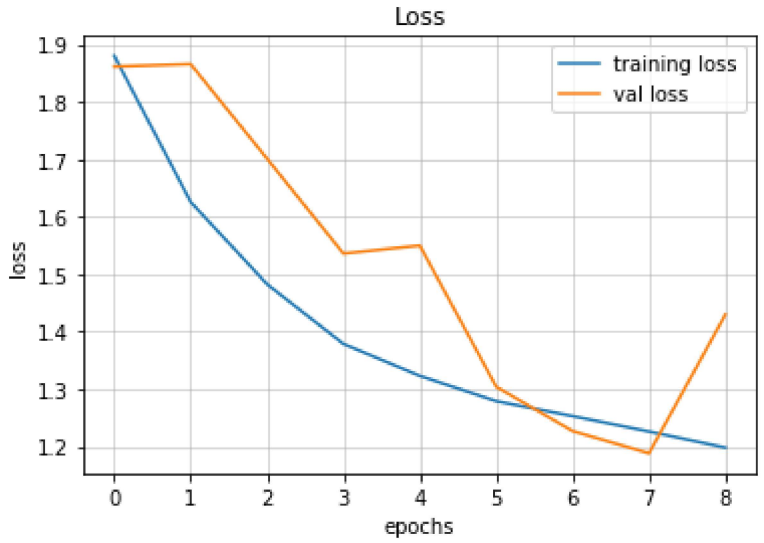
In [17]:
```
# number of epochs to train the NN
history = model.fit(train_generator,
                    steps_per_epoch = train_generator.n//train_generator.batch_size,
                    epochs = 10,
                    validation_data = test_generator,
                    validation_steps = test_generator.n//test_generator.batch_size ,
                    callbacks = [es,]
                   )
```

```
Epoch 1/10
112/112 [==============================] - 232s 2s/step - loss: 1.8799 - accuracy: 0.2558 - val_loss: 1.8612 - val_ac
curacy: 0.1759
Epoch 2/10
112/112 [==============================] - 243s 2s/step - loss: 1.6257 - accuracy: 0.3656 - val_loss: 1.8655 - val_ac
curacy: 0.1787
Epoch 3/10
112/112 [==============================] - 243s 2s/step - loss: 1.4828 - accuracy: 0.4264 - val_loss: 1.7011 - val_ac
curacy: 0.2873
Epoch 4/10
112/112 [==============================] - 247s 2s/step - loss: 1.3788 - accuracy: 0.4671 - val_loss: 1.5360 - val_ac
curacy: 0.4316
Epoch 5/10
112/112 [==============================] - 237s 2s/step - loss: 1.3234 - accuracy: 0.4929 - val_loss: 1.5499 - val_ac
curacy: 0.4363
Epoch 6/10
112/112 [==============================] - 357s 3s/step - loss: 1.2793 - accuracy: 0.5104 - val_loss: 1.3044 - val_ac
curacy: 0.5025
Epoch 7/10
112/112 [==============================] - 260s 2s/step - loss: 1.2533 - accuracy: 0.5199 - val_loss: 1.2273 - val_ac
curacy: 0.5263
Epoch 8/10
112/112 [==============================] - 253s 2s/step - loss: 1.2265 - accuracy: 0.5317 - val_loss: 1.1886 - val_ac
curacy: 0.5587
Epoch 9/10
112/112 [==============================] - 501s 4s/step - loss: 1.1984 - accuracy: 0.5446 - val_loss: 1.4300 - val_ac
curacy: 0.4769
```

In [18]:
```
# plotting graphs for accuracy

plt.plot(history.history['accuracy'], label = 'training accuracy')
plt.plot(history.history['val_accuracy'], label = 'val accuracy')

plt.xlabel('epochs')
plt.ylabel('accuracy')

plt.title('Accuracy')

plt.grid(alpha = 0.6)
plt.legend()
plt.show()
```



In [19]:
```
# plotting for loss

plt.plot(history.history['loss'], label = 'training loss')
plt.plot(history.history['val_loss'], label = 'val loss')

plt.title('Loss')

plt.xlabel('epochs')
plt.ylabel('loss')

plt.grid(alpha = 0.6)
plt.legend()
plt.show()
```



In [20]:
```
pred = model.predict(test_generator)

pred_labels = np.argmax(pred, axis = -1)
```

In [21]:
```
print(classification_report(test_generator.classes , pred_labels ))
```

```
              precision    recall  f1-score   support

           0       0.48      0.34      0.40       960
           1       0.23      0.33      0.27       111
           2       0.38      0.28      0.32      1018
           3       0.82      0.76      0.79      1825
           4       0.38      0.72      0.50      1216
           5       0.49      0.24      0.32      1139
           6       0.63      0.72      0.67       797

    accuracy                           0.53      7066
   macro avg       0.49      0.48      0.47      7066
weighted avg       0.55      0.53      0.52      7066
```

Model 3 - Transfer Learning

In [28]:
```
batch_size = 256

train_datagen_transfer = ImageDataGenerator(rescale = 1.0/255.0,
                                    width_shift_range = 0.1,
                                    height_shift_range = 0.1,
                                    rotation_range = 20,
                                    horizontal_flip = True)


train_generator_transfer = train_datagen_transfer.flow_from_directory(train_folder_path,
                                                      target_size = (128,128),
                                                      color_mode = "rgb",
                                                      batch_size = batch_size,
                                                      class_mode = 'categorical',
                                                      shuffle = True)

test_datagen_transfer = ImageDataGenerator(rescale= 1.0/255)

test_generator_transfer = test_datagen_transfer.flow_from_directory(test_folder_path,
                                                    target_size = (128,128),
                                                    color_mode = "rgb",
                                                    batch_size = batch_size,
                                                    class_mode = 'categorical',
                                                    shuffle = False)
```

```
Found 28820 images belonging to 7 classes.
Found 7066 images belonging to 7 classes.
```

In [29]:

```
#Loading the Mobilenet model
featurizer = MobileNet(include_top = False, weights = 'imagenet', input_shape = (128,128,3))

#Since we have 7 types of expressions, we'll set the nulber of classes to 7

#Adding some layers to the feturizer
x = Flatten()(featurizer.output)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.2)(x)
x = BatchNormalization()(x)
predictions = Dense(7, activation = 'softmax')(x)

model_transfer = Model(inputs = featurizer.input, outputs = predictions)

model_transfer.compile(optimizer = 'Adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])

model_transfer.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobilenet_1_0_128_tf_no_
top.h5
17227776/17225924 [==============================] - 2s 0us/step
17235968/17225924 [==============================] - 2s 0us/step
Model: "model_2"
```

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 128, 128, 3)]	0
conv1 (Conv2D)	(None, 64, 64, 32)	864
conv1_bn (BatchNormalization	(None, 64, 64, 32)	128
conv1_relu (ReLU)	(None, 64, 64, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 64, 64, 32)	288
conv_dw_1_bn (BatchNormaliza	(None, 64, 64, 32)	128
conv_dw_1_relu (ReLU)	(None, 64, 64, 32)	0
conv_pw_1 (Conv2D)	(None, 64, 64, 64)	2048
conv_pw_1_bn (BatchNormaliza	(None, 64, 64, 64)	256
conv_pw_1_relu (ReLU)	(None, 64, 64, 64)	0
conv_pad_2 (ZeroPadding2D)	(None, 65, 65, 64)	0
conv_dw_2 (DepthwiseConv2D)	(None, 32, 32, 64)	576
conv_dw_2_bn (BatchNormaliza	(None, 32, 32, 64)	256
conv_dw_2_relu (ReLU)	(None, 32, 32, 64)	0
conv_pw_2 (Conv2D)	(None, 32, 32, 128)	8192
conv_pw_2_bn (BatchNormaliza	(None, 32, 32, 128)	512
conv_pw_2_relu (ReLU)	(None, 32, 32, 128)	0

conv_dw_3 (DepthwiseConv2D)	(None, 32, 32, 128)	1152
conv_dw_3_bn (BatchNormaliza	(None, 32, 32, 128)	512
conv_dw_3_relu (ReLU)	(None, 32, 32, 128)	0
conv_pw_3 (Conv2D)	(None, 32, 32, 128)	16384
conv_pw_3_bn (BatchNormaliza	(None, 32, 32, 128)	512
conv_pw_3_relu (ReLU)	(None, 32, 32, 128)	0
conv_pad_4 (ZeroPadding2D)	(None, 33, 33, 128)	0
conv_dw_4 (DepthwiseConv2D)	(None, 16, 16, 128)	1152
conv_dw_4_bn (BatchNormaliza	(None, 16, 16, 128)	512
conv_dw_4_relu (ReLU)	(None, 16, 16, 128)	0
conv_pw_4 (Conv2D)	(None, 16, 16, 256)	32768
conv_pw_4_bn (BatchNormaliza	(None, 16, 16, 256)	1024
conv_pw_4_relu (ReLU)	(None, 16, 16, 256)	0
conv_dw_5 (DepthwiseConv2D)	(None, 16, 16, 256)	2304
conv_dw_5_bn (BatchNormaliza	(None, 16, 16, 256)	1024
conv_dw_5_relu (ReLU)	(None, 16, 16, 256)	0
conv_pw_5 (Conv2D)	(None, 16, 16, 256)	65536
conv_pw_5_bn (BatchNormaliza	(None, 16, 16, 256)	1024
conv_pw_5_relu (ReLU)	(None, 16, 16, 256)	0
conv_pad_6 (ZeroPadding2D)	(None, 17, 17, 256)	0
conv_dw_6 (DepthwiseConv2D)	(None, 8, 8, 256)	2304

conv_dw_6_bn (BatchNormaliza	(None, 8, 8, 256)	1024
conv_dw_6_relu (ReLU)	(None, 8, 8, 256)	0
conv_pw_6 (Conv2D)	(None, 8, 8, 512)	131072
conv_pw_6_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_pw_6_relu (ReLU)	(None, 8, 8, 512)	0
conv_dw_7 (DepthwiseConv2D)	(None, 8, 8, 512)	4608
conv_dw_7_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_dw_7_relu (ReLU)	(None, 8, 8, 512)	0
conv_pw_7 (Conv2D)	(None, 8, 8, 512)	262144
conv_pw_7_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_pw_7_relu (ReLU)	(None, 8, 8, 512)	0
conv_dw_8 (DepthwiseConv2D)	(None, 8, 8, 512)	4608
conv_dw_8_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_dw_8_relu (ReLU)	(None, 8, 8, 512)	0
conv_pw_8 (Conv2D)	(None, 8, 8, 512)	262144
conv_pw_8_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_pw_8_relu (ReLU)	(None, 8, 8, 512)	0
conv_dw_9 (DepthwiseConv2D)	(None, 8, 8, 512)	4608
conv_dw_9_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_dw_9_relu (ReLU)	(None, 8, 8, 512)	0
conv_pw_9 (Conv2D)	(None, 8, 8, 512)	262144

conv_pw_9_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_pw_9_relu (ReLU)	(None, 8, 8, 512)	0
conv_dw_10 (DepthwiseConv2D)	(None, 8, 8, 512)	4608
conv_dw_10_bn (BatchNormaliz	(None, 8, 8, 512)	2048
conv_dw_10_relu (ReLU)	(None, 8, 8, 512)	0
conv_pw_10 (Conv2D)	(None, 8, 8, 512)	262144
conv_pw_10_bn (BatchNormaliz	(None, 8, 8, 512)	2048
conv_pw_10_relu (ReLU)	(None, 8, 8, 512)	0
conv_dw_11 (DepthwiseConv2D)	(None, 8, 8, 512)	4608
conv_dw_11_bn (BatchNormaliz	(None, 8, 8, 512)	2048
conv_dw_11_relu (ReLU)	(None, 8, 8, 512)	0
conv_pw_11 (Conv2D)	(None, 8, 8, 512)	262144
conv_pw_11_bn (BatchNormaliz	(None, 8, 8, 512)	2048
conv_pw_11_relu (ReLU)	(None, 8, 8, 512)	0
conv_pad_12 (ZeroPadding2D)	(None, 9, 9, 512)	0
conv_dw_12 (DepthwiseConv2D)	(None, 4, 4, 512)	4608
conv_dw_12_bn (BatchNormaliz	(None, 4, 4, 512)	2048
conv_dw_12_relu (ReLU)	(None, 4, 4, 512)	0
conv_pw_12 (Conv2D)	(None, 4, 4, 1024)	524288
conv_pw_12_bn (BatchNormaliz	(None, 4, 4, 1024)	4096
conv_pw_12_relu (ReLU)	(None, 4, 4, 1024)	0

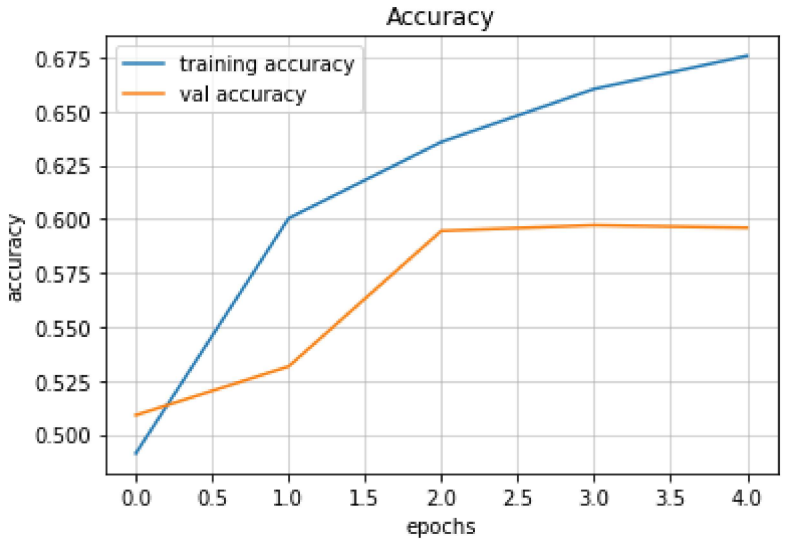
conv_dw_13 (DepthwiseConv2D)	(None, 4, 4, 1024)	9216
conv_dw_13_bn (BatchNormaliz	(None, 4, 4, 1024)	4096
conv_dw_13_relu (ReLU)	(None, 4, 4, 1024)	0
conv_pw_13 (Conv2D)	(None, 4, 4, 1024)	1048576
conv_pw_13_bn (BatchNormaliz	(None, 4, 4, 1024)	4096
conv_pw_13_relu (ReLU)	(None, 4, 4, 1024)	0
flatten_4 (Flatten)	(None, 16384)	0
dense_9 (Dense)	(None, 1024)	16778240
dropout_12 (Dropout)	(None, 1024)	0
batch_normalization_12 (Batc	(None, 1024)	4096
dense_10 (Dense)	(None, 7)	7175

Total params: 20,018,375
Trainable params: 19,994,439
Non-trainable params: 23,936

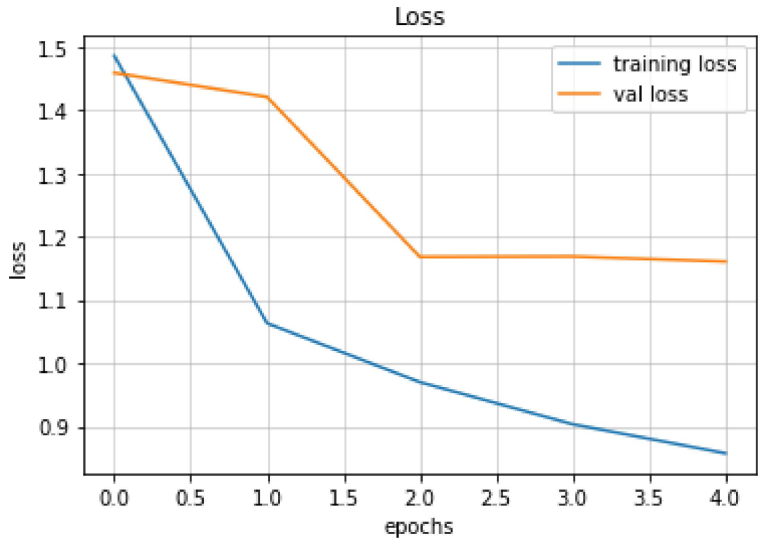
In [30]:
```
history2 = model_transfer.fit(train_generator_transfer,
                              steps_per_epoch = train_generator_transfer.n//train_generator_transfer.batch_size,
                              epochs = 15,
                              validation_data = test_generator_transfer,
                              validation_steps = test_generator_transfer.n//test_generator_transfer.batch_size ,
                              callbacks = [es,]
                              )
```

```
Epoch 1/15
112/112 [==============================] - 876s 8s/step - loss: 1.4857 - accuracy: 0.4912 - val_loss: 1.4588 - val_ac
curacy: 0.5090
Epoch 2/15
112/112 [==============================] - 742s 7s/step - loss: 1.0632 - accuracy: 0.6003 - val_loss: 1.4208 - val_ac
curacy: 0.5315
Epoch 3/15
112/112 [==============================] - 745s 7s/step - loss: 0.9701 - accuracy: 0.6357 - val_loss: 1.1680 - val_ac
curacy: 0.5945
Epoch 4/15
112/112 [==============================] - 741s 7s/step - loss: 0.9036 - accuracy: 0.6603 - val_loss: 1.1686 - val_ac
curacy: 0.5971
Epoch 5/15
112/112 [==============================] - 739s 7s/step - loss: 0.8576 - accuracy: 0.6756 - val_loss: 1.1605 - val_ac
curacy: 0.5959
```

In [32]:
```
# plotting graphs for accuracy

plt.plot(history2.history['accuracy'], label = 'training accuracy')
plt.plot(history2.history['val_accuracy'], label = 'val accuracy')

plt.xlabel('epochs')
plt.ylabel('accuracy')

plt.title('Accuracy')

plt.grid(alpha = 0.6)
plt.legend()
plt.show()
```



In [34]:
```
# plotting for loss

plt.plot(history2.history['loss'], label = 'training loss')
plt.plot(history2.history['val_loss'], label = 'val loss')

plt.title('Loss')

plt.xlabel('epochs')
plt.ylabel('loss')

plt.grid(alpha = 0.6)
plt.legend()
plt.show()
```



In [35]:
```
pred = model_transfer.predict(test_generator_transfer)

pred_labels = np.argmax(pred, axis = -1)
```

In [36]:
```
print(classification_report(test_generator_transfer.classes , pred_labels ))
```

```
              precision    recall  f1-score   support

           0       0.60      0.44      0.51       960
           1       0.89      0.07      0.13       111
           2       0.66      0.12      0.21      1018
           3       0.76      0.89      0.82      1825
           4       0.46      0.81      0.58      1216
           5       0.51      0.37      0.43      1139
           6       0.63      0.80      0.70       797

    accuracy                           0.60      7066
   macro avg       0.64      0.50      0.48      7066
weighted avg       0.62      0.60      0.56      7066
```

Final

- Kernel size could be changed and number of neurons can be changed in the layers
- Different transfer learning model can be used , maybe we can train the whole architecture
- More layers can be added and different activation function can be used