Convolutional Neural Networks for Image Classification

The goal of this project is to build a deep neural network that can recognize objects in photographs and to recognize new objects.

1- Exploring the data

In [55]:
```
# print the first 9 images
print('Train: X=%s, y=%s' % (x_train.shape, y_train.shape))
print('Test: X=%s, y=%s' % (x_test.shape, y_test.shape))

for i in range(9):
    pyplot.subplot(330 + 1 + i)
    pyplot.imshow(x_train[i])
# show the figure
pyplot.show()
```

```
Train: X=(50000, 32, 32, 3), y=(50000, 10)
Test: X=(10000, 32, 32, 3), y=(10000, 10)
```

2- Building Convolutional Neural Network

- CCN layers to find pattern in images,
- Max pooling to reduce the images size
- Dropout layers to prevent neural networks from overfitting

In [5]:
```
# create a model and add layers
model = Sequential()

model.add(Conv2D (32, (3, 3), padding="same", activation = "relu" , input_shape= ( 32, 32, 3)))# (R-G-B)
model.add(Conv2D(32 , (3, 3), activation = "relu"))

model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64 , (3, 3), padding = "same", activation = "relu"))
model.add(Conv2D(64 , (3, 3), activation = "relu"))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(512 , activation = "relu", ))
model.add(Dropout(0.5))

model.add(Dense(10, activation = "softmax"))
```

```
2021-12-12 14:37:05.232737: I tensorflow/core/platform/cpu_feature_guard.cc:145] This TensorFlow binary is optimized with Intel (R) MKL-DNN to use the following CPU instructions in performance critical operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in non-MKL-DNN operations, rebuild TensorFlow with the appropriate compiler flags.
2021-12-12 14:37:05.233643: I tensorflow/core/common_runtime/process_util.cc:115] Creating new thread pool with default inter op setting: 4. Tune using inter_op_parallelism_threads for best performance.
```

In [7]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 30, 30, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1180160
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 1,250,858
Trainable params: 1,250,858
Non-trainable params: 0
_________________________________________________________________
```

3- Training and validation accuracy

In [8]:
```
model.fit( x_train
          , y_train
          , batch_size = 64
          , epochs=30
          , validation_data=(x_test, y_test ), shuffle = True)
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/30
50000/50000 [==============================] - 120s 2ms/step - loss: 1.5509 - accuracy: 0.4307 - val_loss: 1.2244 - val_accurac
y: 0.5598
Epoch 2/30
50000/50000 [==============================] - 111s 2ms/step - loss: 1.1699 - accuracy: 0.5844 - val_loss: 0.9730 - val_accurac
y: 0.6558
Epoch 3/30
50000/50000 [==============================] - 115s 2ms/step - loss: 0.9958 - accuracy: 0.6494 - val_loss: 0.9672 - val_accurac
y: 0.6618
Epoch 4/30
50000/50000 [==============================] - 105s 2ms/step - loss: 0.8929 - accuracy: 0.6865 - val_loss: 0.8156 - val_accurac
y: 0.7193
Epoch 5/30
50000/50000 [==============================] - 145s 3ms/step - loss: 0.8220 - accuracy: 0.7111 - val_loss: 0.7969 - val_accurac
y: 0.7256
Epoch 6/30
50000/50000 [==============================] - 134s 3ms/step - loss: 0.7718 - accuracy: 0.7314 - val_loss: 0.7198 - val_accurac
y: 0.7524
Epoch 7/30
50000/50000 [==============================] - 125s 3ms/step - loss: 0.7264 - accuracy: 0.7443 - val_loss: 0.7104 - val_accurac
y: 0.7498
Epoch 8/30
50000/50000 [==============================] - 128s 3ms/step - loss: 0.6962 - accuracy: 0.7567 - val_loss: 0.6758 - val_accurac
y: 0.7638
Epoch 9/30
50000/50000 [==============================] - 119s 2ms/step - loss: 0.6666 - accuracy: 0.7658 - val_loss: 0.6791 - val_accurac
y: 0.7692
Epoch 10/30
50000/50000 [==============================] - 113s 2ms/step - loss: 0.6485 - accuracy: 0.7739 - val_loss: 0.6807 - val_accurac
y: 0.7660
Epoch 11/30
50000/50000 [==============================] - 141s 3ms/step - loss: 0.6128 - accuracy: 0.7837 - val_loss: 0.6388 - val_accurac
y: 0.7808
Epoch 12/30
50000/50000 [==============================] - 135s 3ms/step - loss: 0.5980 - accuracy: 0.7897 - val_loss: 0.6531 - val_accurac
y: 0.7770
Epoch 13/30
50000/50000 [==============================] - 117s 2ms/step - loss: 0.5818 - accuracy: 0.7932 - val_loss: 0.6443 - val_accurac
y: 0.7831
Epoch 14/30
50000/50000 [==============================] - 110s 2ms/step - loss: 0.5570 - accuracy: 0.8032 - val_loss: 0.6198 - val_accurac
y: 0.7898
Epoch 15/30
50000/50000 [==============================] - 109s 2ms/step - loss: 0.5443 - accuracy: 0.8070 - val_loss: 0.6347 - val_accurac
y: 0.7858
Epoch 16/30
50000/50000 [==============================] - 108s 2ms/step - loss: 0.5293 - accuracy: 0.8131 - val_loss: 0.6075 - val_accurac
y: 0.7919
Epoch 17/30
50000/50000 [==============================] - 108s 2ms/step - loss: 0.5203 - accuracy: 0.8170 - val_loss: 0.6128 - val_accurac
y: 0.7902
Epoch 18/30
50000/50000 [==============================] - 107s 2ms/step - loss: 0.5109 - accuracy: 0.8207 - val_loss: 0.6292 - val_accurac
y: 0.7923
Epoch 19/30
50000/50000 [==============================] - 199s 4ms/step - loss: 0.4966 - accuracy: 0.8273 - val_loss: 0.6328 - val_accurac
y: 0.7912
Epoch 20/30
50000/50000 [==============================] - 108s 2ms/step - loss: 0.4846 - accuracy: 0.8297 - val_loss: 0.6150 - val_accurac
y: 0.7964
Epoch 21/30
50000/50000 [==============================] - 108s 2ms/step - loss: 0.4720 - accuracy: 0.8330 - val_loss: 0.6165 - val_accurac
y: 0.7930
Epoch 22/30
50000/50000 [==============================] - 109s 2ms/step - loss: 0.4626 - accuracy: 0.8349 - val_loss: 0.6276 - val_accurac
y: 0.7968
Epoch 23/30
50000/50000 [==============================] - 108s 2ms/step - loss: 0.4609 - accuracy: 0.8372 - val_loss: 0.6033 - val_accurac
y: 0.7983
Epoch 24/30
50000/50000 [==============================] - 108s 2ms/step - loss: 0.4434 - accuracy: 0.8419 - val_loss: 0.6247 - val_accurac
y: 0.7944
Epoch 25/30
50000/50000 [==============================] - 110s 2ms/step - loss: 0.4454 - accuracy: 0.8422 - val_loss: 0.6344 - val_accurac
y: 0.7939
Epoch 26/30
50000/50000 [==============================] - 114s 2ms/step - loss: 0.4319 - accuracy: 0.8461 - val_loss: 0.6483 - val_accurac
y: 0.7916
Epoch 27/30
50000/50000 [==============================] - 110s 2ms/step - loss: 0.4280 - accuracy: 0.8486 - val_loss: 0.6248 - val_accurac
y: 0.7927
Epoch 28/30
50000/50000 [==============================] - 109s 2ms/step - loss: 0.4183 - accuracy: 0.8512 - val_loss: 0.6275 - val_accurac
y: 0.8004
Epoch 29/30
50000/50000 [==============================] - 111s 2ms/step - loss: 0.4108 - accuracy: 0.8547 - val_loss: 0.6352 - val_accurac
y: 0.7971
Epoch 30/30
50000/50000 [==============================] - 115s 2ms/step - loss: 0.4043 - accuracy: 0.8540 - val_loss: 0.6335 - val_accurac
y: 0.7966
```

Out[8]:
```
<keras.callbacks.callbacks.History at 0x7fa60829bc90>
```