In [78]:
```
import IPython.display as display
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
import os
import pathlib
import scipy.io
import tensorflow as tf

from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, Max
Pooling2D, AveragePooling2D
```

Load Data

In [26]:
```
NUM_TRAIN_SAMPLES = 73257
NUM_TEST_SAMPLES = 26032
IMAGE_SIZE = 32
RGB = 3
```

In [27]:
```
train = scipy.io.loadmat('data/mat/train_32x32.mat')
X_train = np.asarray([train['X'][:,:,:,i] for i in range(NUM_TRAIN_SAMPL
ES)])
y_train = train['y']
```

In [28]:
```
print ('X train: {0} y train: {1}'.format(X_train.shape, y_train.shape))
```

```
X train: (73257, 32, 32, 3) y train: (73257, 1)
```

In [29]:
```
test = scipy.io.loadmat('data/mat/test_32x32.mat')
X_test = np.asarray([test['X'][:,:,:,i] for i in range(NUM_TEST_SAMPLES
)])
y_test = test['y']
```

In [30]:
```
print ('X test: {0} y test: {1}'.format(X_test.shape, y_test.shape))
```

```
X test: (26032, 32, 32, 3) y test: (26032, 1)
```

In [31]:
```
# Normalize pixel values to be between 0 and 1
X_train, X_test = X_train / 255.0, X_test / 255.0
```

In [32]:
```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(X_train[i], cmap=plt.cm.binary)
plt.show()
```

Define Architecture

3-Layer CNN with 3x3 filter size and 32, 64, and 128 filters.

Alternating 2x2 average pooling layers between each convolution

ReLu activation

One softmax and output layer

In [92]:
```
model = tf.keras.Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(IMAGE_SIZE, I
MAGE_SIZE, 3)),
        AveragePooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        AveragePooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu'),
        AveragePooling2D((2, 2))
    ])
```

In [93]:
```
model.summary()
```

```
Model: "sequential_23"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_139 (Conv2D)          (None, 30, 30, 32)        896
_________________________________________________________________
average_pooling2d_6 (Average (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_140 (Conv2D)          (None, 13, 13, 64)        18496
_________________________________________________________________
average_pooling2d_7 (Average (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_141 (Conv2D)          (None, 4, 4, 128)         73856
_________________________________________________________________
average_pooling2d_8 (Average (None, 2, 2, 128)         0
=================================================================
Total params: 93,248
Trainable params: 93,248
Non-trainable params: 0
_________________________________________________________________
```

In [94]:
```
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(11))
```

In [95]:
```
model.summary()
```

```
Model: "sequential_23"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_139 (Conv2D)          (None, 30, 30, 32)        896
_________________________________________________________________
average_pooling2d_6 (Average (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_140 (Conv2D)          (None, 13, 13, 64)        18496
_________________________________________________________________
average_pooling2d_7 (Average (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_141 (Conv2D)          (None, 4, 4, 128)         73856
_________________________________________________________________
average_pooling2d_8 (Average (None, 2, 2, 128)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_12 (Dense)             (None, 64)                32832
_________________________________________________________________
dense_13 (Dense)             (None, 11)                715
=================================================================
Total params: 126,795
Trainable params: 126,795
Non-trainable params: 0
_________________________________________________________________
```

In [96]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_lo
gits=True),
              metrics=['accuracy'])
```

Train Model for 10 epochs

In [97]:
```
history = model.fit(X_train, y_train, epochs=10,
                    validation_data=(X_test, y_test))
```

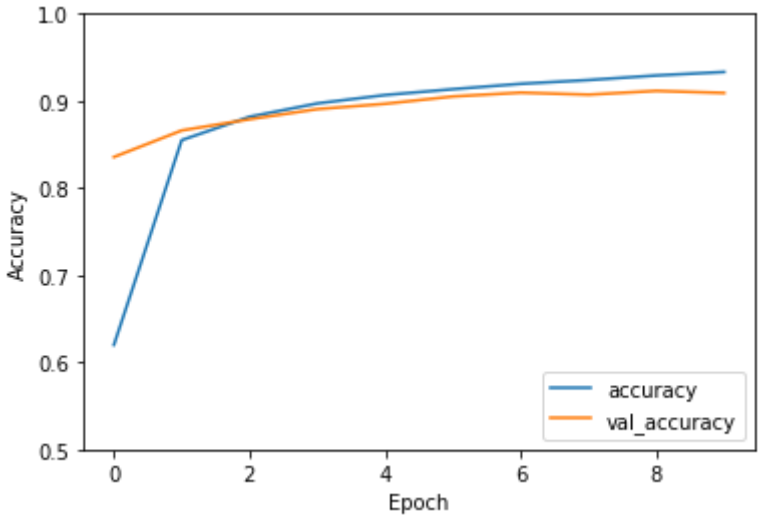
```
Train on 73257 samples, validate on 26032 samples
Epoch 1/10
73257/73257 [==============================] - 28s 383us/sample - loss:
1.1406 - accuracy: 0.6199 - val_loss: 0.5682 - val_accuracy: 0.8356
Epoch 2/10
73257/73257 [==============================] - 29s 395us/sample - loss:
0.4902 - accuracy: 0.8549 - val_loss: 0.4638 - val_accuracy: 0.8661
Epoch 3/10
73257/73257 [==============================] - 29s 393us/sample - loss:
0.4001 - accuracy: 0.8818 - val_loss: 0.4176 - val_accuracy: 0.8789
Epoch 4/10
73257/73257 [==============================] - 28s 388us/sample - loss:
0.3494 - accuracy: 0.8971 - val_loss: 0.3764 - val_accuracy: 0.8906
Epoch 5/10
73257/73257 [==============================] - 27s 373us/sample - loss:
0.3159 - accuracy: 0.9067 - val_loss: 0.3677 - val_accuracy: 0.8967
Epoch 6/10
73257/73257 [==============================] - 27s 373us/sample - loss:
0.2923 - accuracy: 0.9135 - val_loss: 0.3365 - val_accuracy: 0.9051
Epoch 7/10
73257/73257 [==============================] - 27s 373us/sample - loss:
0.2726 - accuracy: 0.9196 - val_loss: 0.3237 - val_accuracy: 0.9095
Epoch 8/10
73257/73257 [==============================] - 27s 371us/sample - loss:
0.2571 - accuracy: 0.9239 - val_loss: 0.3309 - val_accuracy: 0.9072
Epoch 9/10
73257/73257 [==============================] - 27s 370us/sample - loss:
0.2414 - accuracy: 0.9292 - val_loss: 0.3179 - val_accuracy: 0.9115
Epoch 10/10
73257/73257 [==============================] - 28s 379us/sample - loss:
0.2279 - accuracy: 0.9332 - val_loss: 0.3280 - val_accuracy: 0.9091
```

Evaluate Model - 90% accuracy

In [98]:
```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(X_test,  y_test, verbose=2)
```

```
26032/26032 - 3s - loss: 0.3280 - accuracy: 0.9091
```



In [99]:
```
test_acc
```

Out[99]:
```
0.90911186
```

Review

1. Normalized images during pre-processing
2. Experimented with varying number of layers and found that a three-layer convolutional network alternating pooling layers provided the best results
3. Experimented with 32, 64, and 128 filter sizes to extract more detail from each layer
4. Applied a 3x3 filter size for each convolutional layer
5. Used an average pooling layer between each convolution because contrast between house numbers and backgrounds were not clearly defined
6. Used ReLu activation function to quickly train the model
7. Achieved 90% accuracy

In [ ]:
```
```