```
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

model = Sequential()
from keras.utils.vis_utils import plot_model
model = Sequential()

# Step 1 - Convolution
model.add(Convolution2D(32, 3, 1, input_shape = (150, 150, 3),
                        activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

# Step 2 - Adding some other convolutional and Maxpool layers
model.add(Convolution2D(32, 3, 3, activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Convolution2D(32, 3, 3, activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Convolution2D(32, 3, 3, activation = 'relu'))

# Step 3 - Flattening
model.add(Flatten())

# Step 4 - Full connection
model.add(Dense(output_dim = 128, activation = 'relu'))
model.add(Dense(output_dim = 27, activation = 'sigmoid'))

# Compiling the CNN
model.compile(optimizer = 'adam', loss = 'binary_crossentropy',
              metrics = ['accuracy'])


plot_model(model, to_file='model_plot.png', show_shapes=1, show_layer_names=1)


from keras.preprocessing.image import ImageDataGenerator


# define data augmentation configuration
train_datagen = ImageDataGenerator(featurewise_center=True,
                                   featurewise_std_normalization=True,
                                   zca_whitening=True)

# train generator
train_generator = train_datagen.flow_from_directory(
        '/content/drive/My Drive/Thesis/Figures/Augmented photos/train',
```

```
        target_size=(150, 150),
        batch_size=32,
        class_mode='categorical')

# test generator
test_generator = train_datagen.flow_from_directory(
        '/content/drive/My Drive/Thesis/Figures/Augmented photos/validate',
        target_size=(150, 150),
        batch_size=32,
        class_mode='categorical')


# train model
history = model.fit_generator(
    train_generator,
    steps_per_epoch=10,
    epochs=50,
    validation_data=test_generator, # optional - if used needs to be defined
    validation_steps=10)
```

```
import matplotlib.pyplot as plt
fig = plt.figure()
plt.plot(history.history['val_loss'])
plt.legend(['val_loss'], loc='upper left')
plt.title('validation loss vs epoch')
plt.ylabel('validation loss')
plt.xlabel('Epoch')
```

```
import matplotlib.pyplot as plt
fig = plt.figure()
plt.plot(history.history['val_acc'])
plt.legend(['validation'], loc='lower right')
plt.title('validation accuracy vs epoch')
plt.ylabel('validation accuracy')
plt.xlabel('Epoch')
```