```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os
import numpy as np
import matplotlib.pyplot as plt
```

```
from google.colab import files
files.upload()
! rm -rf ~/.kaggle/
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

Choose Files kaggle.json

- **kaggle.json**(application/json) - 62 bytes, last modified: 4/3/2020 - 100% done

```
Saving kaggle.json to kaggle.json
```

```
! pip install -q kaggle
```

```
!pip uninstall -y kaggle
!pip install --upgrade pip
!pip install kaggle==1.5.6
!kaggle -v
```

```
Uninstalling kaggle-1.5.6:
  Successfully uninstalled kaggle-1.5.6
Collecting pip
  Downloading https://files.pythonhosted.org/packages/54/0c/d01aa759fdc501a58f431eb594a1
     |████████████████████████████████| 1.4MB 2.7MB/s
Installing collected packages: pip
  Found existing installation: pip 19.3.1
    Uninstalling pip-19.3.1:
      Successfully uninstalled pip-19.3.1
Successfully installed pip-20.0.2
Collecting kaggle==1.5.6
  Downloading kaggle-1.5.6.tar.gz (58 kB)
     |████████████████████████████████| 58 kB 1.6 MB/s
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (from k
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from kagg
Requirement already satisfied: python-slugify in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.6/dist-pack
Building wheels for collected packages: kaggle
  Building wheel for kaggle (setup.py) ... done
  Created wheel for kaggle: filename=kaggle-1.5.6-py3-none-any.whl size=72859 sha256=f05
  Stored in directory: /root/.cache/pip/wheels/01/3e/ff/77407ebac3ef71a79b9166a8382aecf8
Successfully built kaggle
Installing collected packages: kaggle
Successfully installed kaggle-1.5.6
Kaggle API 1.5.6
```

```
! kaggle datasets download datamunge/sign-language-mnist
```

```
Downloading sign-language-mnist.zip to /content
 69% 43.0M/62.6M [00:00<00:00, 41.6MB/s]
100% 62.6M/62.6M [00:00<00:00, 88.9MB/s]
```

```
! ls kaggle/input
```

```
american_sign_language.PNG  sign_mnist_test      sign_mnist_train.csv
amer_sign2.png              sign_mnist_test.csv
amer_sign3.png              sign_mnist_train
```

```
! mkdir kaggle/
! mkdir kaggle/input
! unzip sign-language-mnist.zip -d kaggle/input/
```

```
Archive:  sign-language-mnist.zip
  inflating: kaggle/input/amer_sign2.png
  inflating: kaggle/input/amer_sign3.png
  inflating: kaggle/input/american_sign_language.PNG
  inflating: kaggle/input/sign_mnist_test.csv
  inflating: kaggle/input/sign_mnist_test/sign_mnist_test.csv
  inflating: kaggle/input/sign_mnist_train.csv
  inflating: kaggle/input/sign_mnist_train/sign_mnist_train.csv
```

```
! mkdir train
! unzip dogs-vs-cats/train -d train
! mkdir test
! unzip dogs-vs-cats/test1 -d test
```

Задание 1. Загрузите данные. Разделите исходный набор данных на обучающую и валидацио

```
import csv

import os
for dirname, _, filenames in os.walk('kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

def get_data(filename):
    labels = []
    images = []
    with open(filename) as training_file:
        csv_reader = csv.reader(training_file)
        next(csv_reader)
        for row in csv_reader:
            labels.append(row[0])
            pixels = row[1:785]
            pixels_as_array = np.array_split(pixels, 28)
            images.append(pixels_as_array)
        labels = np.array(labels).astype('float')
        images = np.array(images).astype('float')
    return images, labels


training_images, training_labels = get_data('kaggle/input/sign_mnist_train/sign_mnist_tr
testing_images, testing_labels = get_data('kaggle/input/sign_mnist_test/sign_mnist_test.

print(training_images.shape)
print(training_labels.shape)
print(testing_images.shape)
print(testing_labels.shape)

# Their output should be:
# (27455, 28, 28)
# (27455,)
# (27455,)
# (7172, 28, 28)
# (7172,)
```

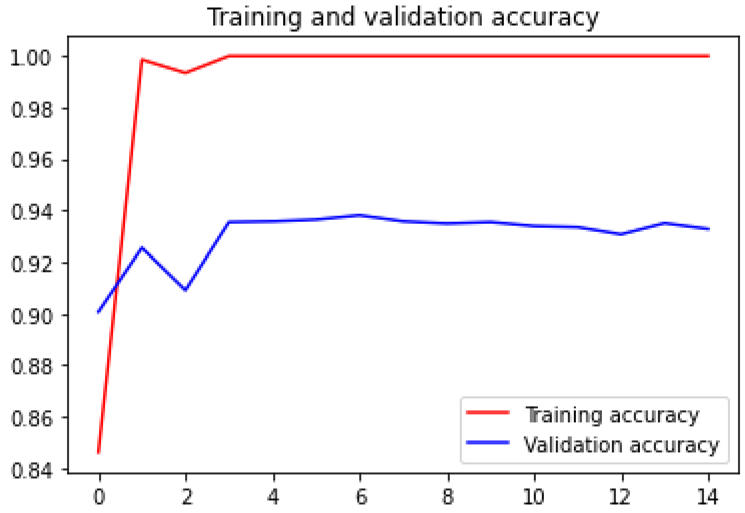
```
kaggle/input/sign_mnist_train.csv
kaggle/input/amer_sign3.png
kaggle/input/sign_mnist_test.csv
kaggle/input/amer_sign2.png
kaggle/input/american_sign_language.PNG
kaggle/input/sign_mnist_train/sign_mnist_train.csv
kaggle/input/sign_mnist_test/sign_mnist_test.csv
(27455, 28, 28)
(27455,)
(7172, 28, 28)
(7172,)
```

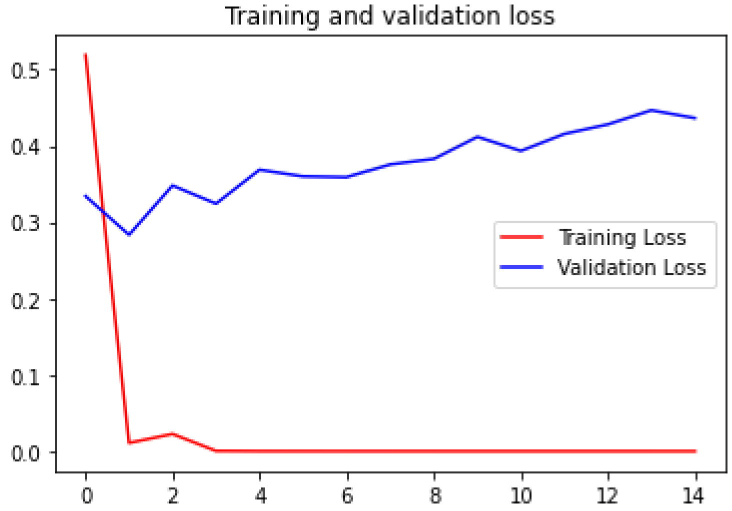
Задание 2. Реализуйте глубокую нейронную сеть со сверточными слоями. Какое качество кла архитектура сети была использована?

```
training_images, training_labels = get_data('kaggle/input/sign_mnist_train/sign_mnist_tr
testing_images, testing_labels = get_data('kaggle/input/sign_mnist_test/sign_mnist_test.

print(training_images.shape)
print(training_labels.shape)
print(testing_images.shape)
print(testing_labels.shape)

from tensorflow.keras.preprocessing.image import ImageDataGenerator

training_images = np.expand_dims(training_images, axis=3)
testing_images = np.expand_dims(testing_images, axis=3)
train_datagen = ImageDataGenerator(
    rescale=1./255.,
)
validation_datagen = ImageDataGenerator(
    rescale=1./255.
)

print(training_images.shape)
print(testing_images.shape)
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', input_shape=(28,28,1)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dense(26, activation='softmax')
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    optimizer=tf.keras.optimizers.Adam(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

len_training_images = len(training_images)
len_testing_images = len(testing_images)

train_flow = train_datagen.flow(training_images, training_labels, batch_size=32)
val_flow = validation_datagen.flow(testing_images, testing_labels, batch_size=32)

history = model.fit_generator(
    train_flow,
    steps_per_epoch=len_training_images/32,
    epochs=15,
    validation_data=val_flow,
    validation_steps=len_testing_images/32
)

model.evaluate(testing_images, testing_labels)
```

```
(27455, 28, 28)
(27455,)
(7172, 28, 28)
(7172,)
(27455, 28, 28, 1)
(7172, 28, 28, 1)
Epoch 1/15
858/857 [==============================] - 4s 5ms/step - loss: 0.5189 - accuracy: 0.8461
Epoch 2/15
858/857 [==============================] - 4s 5ms/step - loss: 0.0109 - accuracy: 0.9986
Epoch 3/15
858/857 [==============================] - 4s 5ms/step - loss: 0.0226 - accuracy: 0.9934
Epoch 4/15
858/857 [==============================] - 4s 5ms/step - loss: 4.0983e-04 - accuracy: 1.
Epoch 5/15
858/857 [==============================] - 4s 5ms/step - loss: 8.6208e-05 - accuracy: 1.
Epoch 6/15
858/857 [==============================] - 4s 5ms/step - loss: 4.6748e-05 - accuracy: 1.
Epoch 7/15
858/857 [==============================] - 4s 5ms/step - loss: 2.7993e-05 - accuracy: 1.
Epoch 8/15
858/857 [==============================] - 4s 5ms/step - loss: 1.7147e-05 - accuracy: 1.
Epoch 9/15
858/857 [==============================] - 4s 5ms/step - loss: 1.1162e-05 - accuracy: 1.
Epoch 10/15
858/857 [==============================] - 4s 5ms/step - loss: 7.2979e-06 - accuracy: 1.
Epoch 11/15
858/857 [==============================] - 4s 5ms/step - loss: 4.7882e-06 - accuracy: 1.
Epoch 12/15
858/857 [==============================] - 4s 5ms/step - loss: 3.0185e-06 - accuracy: 1.
Epoch 13/15
858/857 [==============================] - 4s 5ms/step - loss: 1.9884e-06 - accuracy: 1.
Epoch 14/15
858/857 [==============================] - 4s 5ms/step - loss: 1.2782e-06 - accuracy: 1.
Epoch 15/15
858/857 [==============================] - 4s 5ms/step - loss: 7.8482e-07 - accuracy: 1.
225/225 [==============================] - 1s 3ms/step - loss: 137.8947 - accuracy: 0.89
[137.89474487304688, 0.8969603776931763]
```

```python
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'r', label='Training Loss')
plt.plot(epochs, loss, 'r', label='Training Loss')
plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





```
training_images, training_labels = get_data('kaggle/input/sign_mnist_train/sign_mnist_tr
testing_images, testing_labels = get_data('kaggle/input/sign_mnist_test/sign_mnist_test.

print(training_images.shape)
print(training_labels.shape)
print(testing_images.shape)
print(testing_labels.shape)

from tensorflow.keras.preprocessing.image import ImageDataGenerator

training_images = np.expand_dims(training_images, axis=3)
testing_images = np.expand_dims(testing_images, axis=3)
train_datagen = ImageDataGenerator(
    rescale=1./255.,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
validation_datagen = ImageDataGenerator(
    rescale=1./255.
)

print(training_images.shape)
print(testing_images.shape)

```

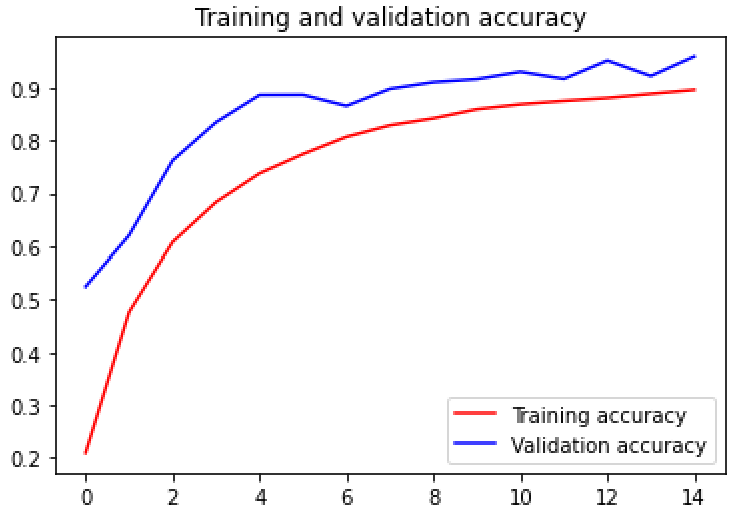
```
(27455, 28, 28, 1)
(7172, 28, 28, 1)
```

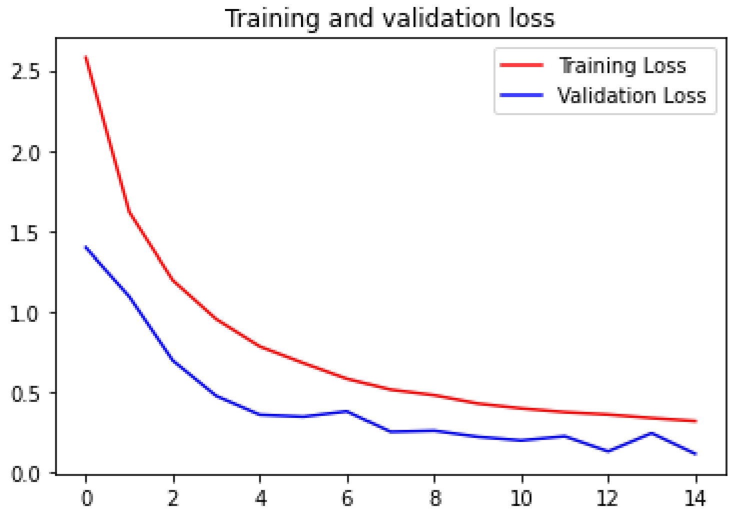
Задание 3. Примените дополнение данных (data augmentation). Как это повлияло на качество

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', input_shape=(28,28,1)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dense(26, activation='softmax')
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

len_training_images = len(training_images)
len_testing_images = len(testing_images)

train_flow = train_datagen.flow(training_images, training_labels, batch_size=32)
val_flow = validation_datagen.flow(testing_images, testing_labels, batch_size=32)

history = model.fit_generator(
    train_flow,
    steps_per_epoch=len_training_images/32,
    epochs=15,
    validation_data=val_flow,
    validation_steps=len_testing_images/32
)

model.evaluate(testing_images, testing_labels)
```

```
WARNING:tensorflow:From <ipython-input-25-af062da55dcb>:30: Model.fit_generator (from te
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/15
858/857 [==============================] - 15s 18ms/step - loss: 2.5796 - accuracy: 0.20
Epoch 2/15
858/857 [==============================] - 14s 17ms/step - loss: 1.6166 - accuracy: 0.47
Epoch 3/15
858/857 [==============================] - 14s 16ms/step - loss: 1.1934 - accuracy: 0.60
Epoch 4/15
858/857 [==============================] - 14s 16ms/step - loss: 0.9515 - accuracy: 0.68
Epoch 5/15
858/857 [==============================] - 14s 17ms/step - loss: 0.7813 - accuracy: 0.73
Epoch 6/15
858/857 [==============================] - 14s 17ms/step - loss: 0.6781 - accuracy: 0.77
Epoch 7/15
858/857 [==============================] - 14s 17ms/step - loss: 0.5802 - accuracy: 0.80
Epoch 8/15
858/857 [==============================] - 14s 16ms/step - loss: 0.5118 - accuracy: 0.82
Epoch 9/15
858/857 [==============================] - 14s 16ms/step - loss: 0.4775 - accuracy: 0.84
Epoch 10/15
858/857 [==============================] - 14s 16ms/step - loss: 0.4254 - accuracy: 0.85
Epoch 11/15
858/857 [==============================] - 14s 17ms/step - loss: 0.3951 - accuracy: 0.86
Epoch 12/15
858/857 [==============================] - 14s 16ms/step - loss: 0.3717 - accuracy: 0.87
Epoch 13/15
858/857 [==============================] - 14s 16ms/step - loss: 0.3569 - accuracy: 0.88
Epoch 14/15
858/857 [==============================] - 14s 16ms/step - loss: 0.3348 - accuracy: 0.88
Epoch 15/15
858/857 [==============================] - 14s 16ms/step - loss: 0.3161 - accuracy: 0.89
225/225 [==============================] - 1s 3ms/step - loss: 121.4452 - accuracy: 0.75
[121.44520568847656, 0.7582264542579651]
```

```python
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'r', label='Training Loss')
plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```

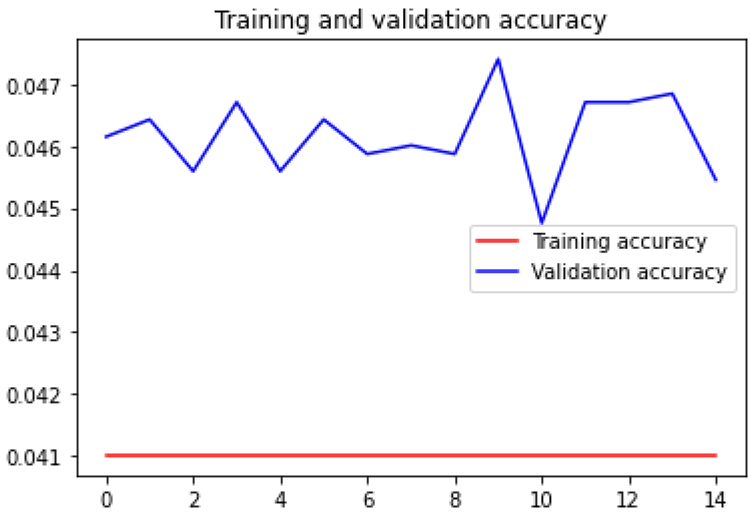


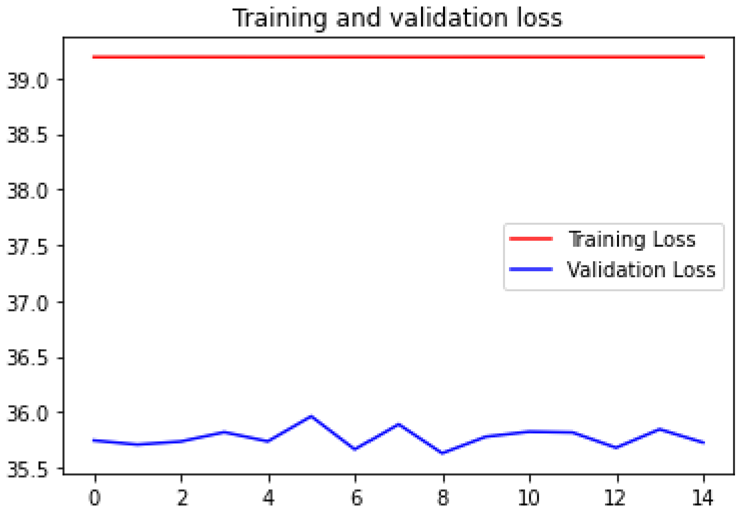


Задание 4. Поэкспериментируйте с готовыми нейронными сетями (например, AlexNet, VGG16, передаточное обучение. Как это повлияло на качество классификатора? Можно ли было обой результат удалось получить на контрольной выборке?

```
from tensorflow.keras.layers import GlobalAveragePooling2D

training_images, training_labels = get_data('kaggle/input/sign_mnist_train/sign_mnist_tr
testing_images, testing_labels = get_data('kaggle/input/sign_mnist_test/sign_mnist_test.

print(training_images.shape)
print(training_labels.shape)
print(testing_images.shape)
print(testing_labels.shape)


def transform(dataset):
  newDataset = list()
  for x in dataset:
    x = np.repeat(x, 3, 2)
    newDataset.append(x)
  return np.array(newDataset)
training_images = np.expand_dims(training_images, axis=3)
testing_images = np.expand_dims(testing_images, axis=3)

testing_images = transform(testing_images)
training_images = transform(training_images)
print(training_images.shape)
print(testing_images.shape)

from tensorflow.keras.preprocessing.image import ImageDataGenerator



train_datagen = ImageDataGenerator(
    rescale=1./255.,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
validation_datagen = ImageDataGenerator(
    rescale=1./255.
)

print(training_images.shape)
print(testing_images.shape)


from tensorflow.keras.applications import MobileNet
from tensorflow.keras.models import Model

base_model=MobileNet(weights='imagenet',include_top=False)

x=base_model.output
x=GlobalAveragePooling2D()(x)
x=Dense(1024,activation='relu')(x)
x=Dense(1024,activation='relu')(x)
x=Dense(512,activation='relu')(x)
preds=Dense(25,activation='softmax')(x)
model=Model(inputs=base_model.input,outputs=preds)
model.summary()
for layer in model.layers:
    layer.trainable=False
model.compile(optimizer='Adam',loss='binary_crossentropy',metrics=['accuracy'])

len_training_images = len(training_images)
len_training_images = len(training_images)
len_testing_images = len(testing_images)

train_flow = train_datagen.flow(training_images, training_labels, batch_size=32)
val_flow = validation_datagen.flow(testing_images, testing_labels, batch_size=32)

history = model.fit_generator(
    train_flow,
    steps_per_epoch=len_training_images/32,
    epochs=15,
    validation_data=val_flow,
    validation_steps=len_testing_images/32
)
```

```
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'r', label='Training Loss')
plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```


Training and validation accuracy
0.047
0.046
0.045
0.044
0.043
0.042
0.041
0
2
4
6
8
10
12
14
Training accuracy
Validation accuracy


Training and validation loss
39.0
38.5
38.0
37.5
37.0
36.5
36.0
35.5
0
2
4
6
8
10
12
14
Training Loss
Validation Loss