trace_neural_network

March 28, 2021

1 Script configuration

[42]:

```
!pip install pandas
!pip install matplotlib
!pip install sklearn
!pip install tensorflow
!pip install pydot
!pip install graphviz

import random
import pandas as pd
import numpy as np
from datetime import datetime

# Make numpy values easier to read.
np.set_printoptions(precision=3, suppress=True)
from matplotlib import pyplot as plt
from IPython.display import clear_output

import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental import preprocessing

tf.random.set_seed(123)

class CSV_FORMATS():
  METRICS = "metrics" # dataset format with extracted metrics
  OPERATIONS = "operations" # dataset format where every entry is an entity operation

class TraceFeature():
  def __init__(self, name: str, first_idx: int, last_idx: int =None):
    self.name = name
    self.first_idx = first_idx
    self.last_idx = last_idx
```

```
Looking in indexes: https://pypi.org/simple,
https://pip:****@pypi.infra.unbabel.com/simple/
Requirement already satisfied: pandas in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(1.2.3)
Requirement already satisfied: pytz>=2017.3 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from pandas) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from pandas) (2.8.1)
Requirement already satisfied: numpy>=1.16.5 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from pandas) (1.19.5)
Requirement already satisfied: six>=1.5 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from python-dateutil>=2.7.3->pandas) (1.15.0)
WARNING: You are using pip version 20.1; however, version 21.0.1 is
available.
You should consider upgrading via the '/usr/local/bin/python3.8 -m pip install
--upgrade pip' command.
Looking in indexes: https://pypi.org/simple,
https://pip:****@pypi.infra.unbabel.com/simple/
Requirement already satisfied: matplotlib in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(3.4.0)
Requirement already satisfied: pillow>=6.2.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from matplotlib) (8.1.2)
Requirement already satisfied: pyparsing>=2.2.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from matplotlib) (2.4.7)
Requirement already satisfied: cycler>=0.10 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>=2.7 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from matplotlib) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from matplotlib) (1.3.1)
Requirement already satisfied: numpy>=1.16 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from matplotlib) (1.19.5)
Requirement already satisfied: six in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from cycler>=0.10->matplotlib) (1.15.0)
```

```
WARNING: You are using pip version 20.1; however, version 21.0.1 is

available.

You should consider upgrading via the '/usr/local/bin/python3.8 -m pip install

--upgrade pip' command.
Looking in indexes: https://pypi.org/simple,
https://pip:****@pypi.infra.unbabel.com/simple/
Requirement already satisfied: sklearn in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(0.0)
Requirement already satisfied: scikit-learn in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from sklearn) (0.24.1)
Requirement already satisfied: scipy>=0.19.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from scikit-learn->sklearn) (1.6.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from scikit-learn->sklearn) (2.1.0)
Requirement already satisfied: joblib>=0.11 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from scikit-learn->sklearn) (1.0.1)
Requirement already satisfied: numpy>=1.13.3 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from scikit-learn->sklearn) (1.19.5)
WARNING: You are using pip version 20.1; however, version 21.0.1 is

available.

You should consider upgrading via the '/usr/local/bin/python3.8 -m pip install

--upgrade pip' command.
Looking in indexes: https://pypi.org/simple,
https://pip:****@pypi.infra.unbabel.com/simple/
Requirement already satisfied: tensorflow in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(2.4.1)
Requirement already satisfied: h5py~=2.10.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (2.10.0)
Requirement already satisfied: six~=1.15.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (1.15.0)
Requirement already satisfied: absl-py~=0.10 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (0.12.0)
Requirement already satisfied: numpy~=1.19.2 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
```

```
(from tensorflow) (1.19.5)
Requirement already satisfied: keras-preprocessing~=1.1.2 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (1.1.2)
Requirement already satisfied: protobuf>=3.9.2 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (3.12.2)
Requirement already satisfied: tensorflow-estimator<2.5.0,>=2.4.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions~=3.7.4 in
/Users/josecorreia/Library/Python/3.8/lib/python/site-packages (from tensorflow)
(3.7.4.3)
Requirement already satisfied: grpcio~=1.32.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (1.32.0)
Requirement already satisfied: google-pasta~=0.2 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (0.2.0)
Requirement already satisfied: opt-einsum~=3.3.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (3.3.0)
Requirement already satisfied: astunparse~=1.6.3 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (1.6.3)
Requirement already satisfied: gast==0.3.3 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (0.3.3)
Requirement already satisfied: tensorboard~=2.4 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (2.4.1)
Requirement already satisfied: termcolor~=1.1.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (1.1.0)
Requirement already satisfied: wrapt~=1.12.1 in
/Users/josecorreia/Library/Python/3.8/lib/python/site-packages (from tensorflow)
(1.12.1)
Requirement already satisfied: flatbuffers~=1.12.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (1.12)
Requirement already satisfied: wheel~=0.35 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorflow) (0.36.2)
Requirement already satisfied: setuptools in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from protobuf>=3.9.2->tensorflow) (41.2.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
```

```
(from tensorboard~=2.4->tensorflow) (1.8.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorboard~=2.4->tensorflow) (0.4.2)
Requirement already satisfied: markdown>=2.6.8 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorboard~=2.4->tensorflow) (3.3.4)
Requirement already satisfied: requests<3,>=2.21.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorboard~=2.4->tensorflow) (2.24.0)
Requirement already satisfied: werkzeug>=0.11.15 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorboard~=2.4->tensorflow) (1.0.1)
Requirement already satisfied: google-auth<2,>=1.6.3 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from tensorboard~=2.4->tensorflow) (1.23.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.4->tensorflow) (1.3.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from requests<3,>=2.21.0->tensorboard~=2.4->tensorflow) (1.25.10)
Requirement already satisfied: chardet<4,>=3.0.2 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from requests<3,>=2.21.0->tensorboard~=2.4->tensorflow) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from requests<3,>=2.21.0->tensorboard~=2.4->tensorflow) (2020.6.20)
Requirement already satisfied: idna<3,>=2.5 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from requests<3,>=2.21.0->tensorboard~=2.4->tensorflow) (2.10)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from google-auth<2,>=1.6.3->tensorboard~=2.4->tensorflow) (4.1.1)
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3.5" in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from google-auth<2,>=1.6.3->tensorboard~=2.4->tensorflow) (4.6)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from google-auth<2,>=1.6.3->tensorboard~=2.4->tensorflow) (0.2.8)
Requirement already satisfied: oauthlib>=3.0.0 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.4->tensorflow) (3.1.0)
Requirement already satisfied: pyasn1>=0.1.3 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from rsa<5,>=3.1.4; python_version >= "3.5"->google-
auth<2,>=1.6.3->tensorboard~=2.4->tensorflow) (0.4.8)
```

```
WARNING: You are using pip version 20.1; however, version 21.0.1 is

available.

You should consider upgrading via the '/usr/local/bin/python3.8 -m pip install

--upgrade pip' command.
Looking in indexes: https://pypi.org/simple,
https://pip:****@pypi.infra.unbabel.com/simple/
Requirement already satisfied: pydot in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(1.4.2)
Requirement already satisfied: pyparsing>=2.1.4 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(from pydot) (2.4.7)
WARNING: You are using pip version 20.1; however, version 21.0.1 is

available.

You should consider upgrading via the '/usr/local/bin/python3.8 -m pip install

--upgrade pip' command.
Looking in indexes: https://pypi.org/simple,
https://pip:****@pypi.infra.unbabel.com/simple/
Requirement already satisfied: graphviz in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages
(0.16)
WARNING: You are using pip version 20.1; however, version 21.0.1 is

available.

You should consider upgrading via the '/usr/local/bin/python3.8 -m pip install

--upgrade pip' command.
```

2 Read dataset

[43]:

```
def read_dataset(file, row_names, rows_to_exclude):
  return pd.read_csv(CSV_FILE, names=CSV_ROWS, skiprows=1, usecols = [i for i in␣
 ↪CSV_ROWS if i not in CSV_ROWS_TO_EXCLUDE])


def split_dataset_by_trace_features(trace_dataset):
  features = []

  for idx, name in enumerate(trace_dataset["Feature"]):
    current_feature = features[len(features) - 1] if len(features) > 0 else None

    if current_feature is None or name != current_feature.name:
      if len(features) > 0:
        last_feature = features[len(features) - 1]
        last_feature.last_idx = idx - 1

      features.append(TraceFeature(name, idx))

  current_feature.last_idx = idx
  return features


def get_empty_features_dict(dataset_features):
  return {name:[] for name, _ in dataset_features.items()}


def create_batch(dataset_features, dataset_labels, trace_features):
  batch_features = get_empty_features_dict(dataset_features)
  batch_labels = None

  for idx, feature in enumerate(trace_features):
    for key, values in dataset_features.items():
      feature_values = values[feature.first_idx:feature.last_idx + 1]
      batch_features[key] = np.concatenate((batch_features[key], feature_values))

    feature_labels = np.asarray(dataset_labels[feature.first_idx:feature.
 ↪last_idx + 1]).astype('float32').reshape((-1,1))

    if idx == 0:
      batch_labels = feature_labels
    else:
      batch_labels = np.concatenate((batch_labels, feature_labels))

  return batch_features, batch_labels


def get_kfold_iteration_batches(
    iteration,
    dataset_features,
    dataset_labels,
    trace_features,
    training_features_size,
    validation_features_size,
    testing_features_size
):
  testing_start_idx = iteration * testing_features_size
  testing_end_idx = testing_start_idx + testing_features_size
  testing_features = trace_features[testing_start_idx:testing_end_idx]

  if iteration == 0:
        training_start_idx = testing_end_idx + validation_features_size
        training_features = trace_features[training_start_idx:]
  elif iteration < (K_FOLD_VALUE - 1):
        training_start_idx_2 = testing_end_idx + validation_features_size
        training_features = trace_features[:testing_start_idx] +␣
↪trace_features[training_start_idx_2:]
  else:
        training_features = trace_features[:testing_start_idx]

  # now we divide the dataset into batches
  training_batch_features, training_batch_labels =␣
↪create_batch(dataset_features, dataset_labels, training_features)
  testing_batch_features, testing_batch_labels = create_batch(dataset_features,␣
↪dataset_labels, testing_features)

  validation_batch_features = None
  validation_batch_labels = None
  if APPLY_FIT_VALIDATION:
    validation_end_idx = testing_end_idx + validation_features_size
    validation_features = trace_features[testing_end_idx:validation_end_idx]
    validation_batch_features, validation_batch_labels =␣
↪create_batch(dataset_features, dataset_labels, validation_features)

  return (training_batch_features, training_batch_labels),␣
↪(testing_batch_features, testing_batch_labels), (validation_batch_features,␣
↪validation_batch_labels)
```

3 Preprocessing data

To build the preprocessing model, start by building a set of symbolic keras.Input objects, matching the names and data-types of the CSV columns.

[44]:
```
def create_input_objects(dataset_features):
  inputs = {}

  for name, column in dataset_features.items():
    dtype = column.dtype
    if dtype == object:
      dtype = tf.string
    else:
      dtype = tf.float32

    inputs[name] = tf.keras.Input(shape=(1,), name=name, dtype=dtype)

  return inputs
```

The first step in the preprocessing logic is to concatenate the numeric inputs together, and run them through a normalization layer:

[45]:
```
def create_preprocessing_logic(dataset, dataset_features, inputs):
  numeric_inputs = {name:input for name,input in inputs.items()
                    if input.dtype==tf.float32}

  preprocessed_inputs = []
  if numeric_inputs:
    x = layers.Concatenate()(list(numeric_inputs.values()))
    norm = preprocessing.Normalization()
    norm.adapt(np.array(dataset[numeric_inputs.keys()]))
    all_numeric_inputs = norm(x)

    # Collect all the symbolic preprocessing results, to concatenate them later.
    preprocessed_inputs = [all_numeric_inputs]

    # For the string inputs use the preprocessing.StringLookup function to map␣
↪from
    # strings to integer indices in a vocabulary. Next, use preprocessing.
↪CategoryEncoding
    # to convert the indexes into float32 data appropriate for the model.
    for name, input in inputs.items():
      if input.dtype == tf.float32:
        continue

      lookup = preprocessing.StringLookup(vocabulary=np.
↪unique(dataset_features[name]))
      one_hot = preprocessing.CategoryEncoding(max_tokens=lookup.vocab_size())

      x = lookup(input)
      x = one_hot(x)
      preprocessed_inputs.append(x)

  preprocessed_inputs_cat = layers.Concatenate()(preprocessed_inputs)
  preprocessing_model = tf.keras.Model(inputs, preprocessed_inputs_cat)
  tf.keras.utils.plot_model(model = preprocessing_model , rankdir="LR", dpi=126,␣
↪show_shapes=True)

  return preprocessing_model
```

4 Design Neural Network model

Now build the model on top of this:

[46]:

```
def build_neural_network_model(body, preprocessing_head, inputs, loss,␣
↪optimizer):
    preprocessed_inputs = preprocessing_head(inputs)
    result = tf.keras.Sequential(body)(preprocessed_inputs)
    model = tf.keras.Model(inputs, result)

    # The purpose of loss functions is to compute the quantity that
    # a model should seek to minimize during training.
    # Binary classification loss function comes into play when solving a problem
    # involving just two classes (1 or 0)

    # Adam optimization is a stochastic gradient descent method that is based on
    # adaptive estimation of first-order and second-order moments.
    # is computationally efficient, has little memory requirement, and is well
    # suited for problems that are large in terms of data/parameters"
    model.compile(
        loss=loss,
        optimizer=optimizer,
        metrics=["accuracy"],
    )
    return model
```

5 Training

[47]:

```
def fit_neural_network(model, training_features, training_labels,␣
↪validation_features, validation_labels, epochs, shuffle, weights):
    callbacks = [
        tf.keras.callbacks.EarlyStopping(
            # Stop training when `loss` is no longer improving
            monitor="loss",
            # "no longer improving" being defined as "no better than 1e-2 less"
            min_delta=1e-4,
            # "no longer improving" being further defined as "for at least 2␣
↪epochs"
            patience=2,
            verbose=1,
        )
    ]

    history = model.fit(
        x=training_features,
        y=training_labels,
        callbacks=callbacks,
        shuffle=shuffle,
        epochs=epochs,
        validation_data=(validation_features, validation_labels) if␣
   ↪APPLY_FIT_VALIDATION else None,
        class_weight=weights, # This argument allows you to define a dictionary␣
   ↪that maps class integer values to the importance to apply to each class.
        verbose=0,
    )

    return history
```

[48]:

```
def plot_training_results(history):
  # plot loss during training
  plt.figure(1)
  plt.title('Loss')
  plt.plot(history.history['loss'], label='train')

  if APPLY_FIT_VALIDATION:
    plt.plot(history.history['val_loss'], label='test')

  plt.legend()

  # plot accuracy during training
  plt.figure(2)
  plt.title('Accuracy')
  plt.plot(history.history['accuracy'], label='train')

  if APPLY_FIT_VALIDATION:
    plt.plot(history.history['val_accuracy'], label='test')

  plt.legend()
  plt.show()

  print(f"\nTraining results:\nFinal loss: {history.history['loss'][len(history.
 ↪history['loss'])-1]}")
  print(f"Final accuracy: {history.history['accuracy'][len(history.
 ↪history['accuracy'])-1]}\n")
```

6 Testing

[60]:

```
from sklearn.metrics import roc_curve, auc

def test_model(model, testing_features, testing_labels, verbose=True):
  print(f"Results for {testing_labels.size} test samples\n")

  results = model.evaluate(testing_features, testing_labels,␣
 ↪batch_size=testing_labels.size, verbose=0)
```

```
  print(f"Loss {results[0]} | Recall: {results[1]}\n")

  predictions = model.predict(testing_features)

  if verbose:
    for idx, prediction in enumerate(predictions):
      label = 0 if prediction[0] > 0.500 else 1
      percentage = prediction[0] if label == 0 else prediction[1]
      percentage = int(percentage * 100)
      correct_label = testing_labels[idx]
      feature = testing_features["Feature"][idx]

      print(f"Prediction: {label} ({percentage} %) | Correct: {correct_label} |␣
 ↪Feature: {feature}")

  return predictions

# !!!!!!!!!!!!!!!!!!
#  If one feature has multiple clusters being the orchestrator, we should select␣
 ↪the one with
#  the highest probability

# evaluate the ROC AUC of the predictions
def plot_testing_results(predictions, testing_labels):
  results = []
  for prediction in predictions:
    label = 0 if prediction[0] > 0.5 else 1
    results.append(label)

  fpr_keras, tpr_keras, thresholds_keras = roc_curve(testing_labels, results)

  auc_keras = auc(fpr_keras, tpr_keras)

  print("\n")
  plt.figure(1)
  plt.plot([0, 1], [0, 1], 'k--')
  plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
  plt.xlabel('False positive rate')
  plt.ylabel('True positive rate')
  plt.title('ROC curve')
  plt.legend(loc='best')
  plt.show()

  print(f"AUC: {auc_keras}")
  return auc_keras
```

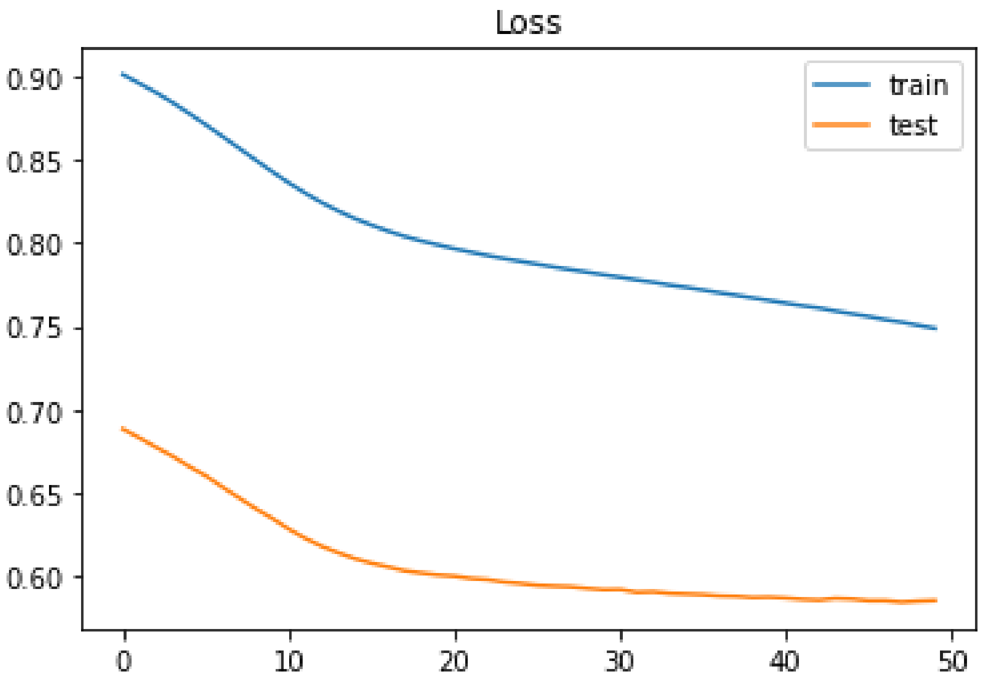
7 Main script execution

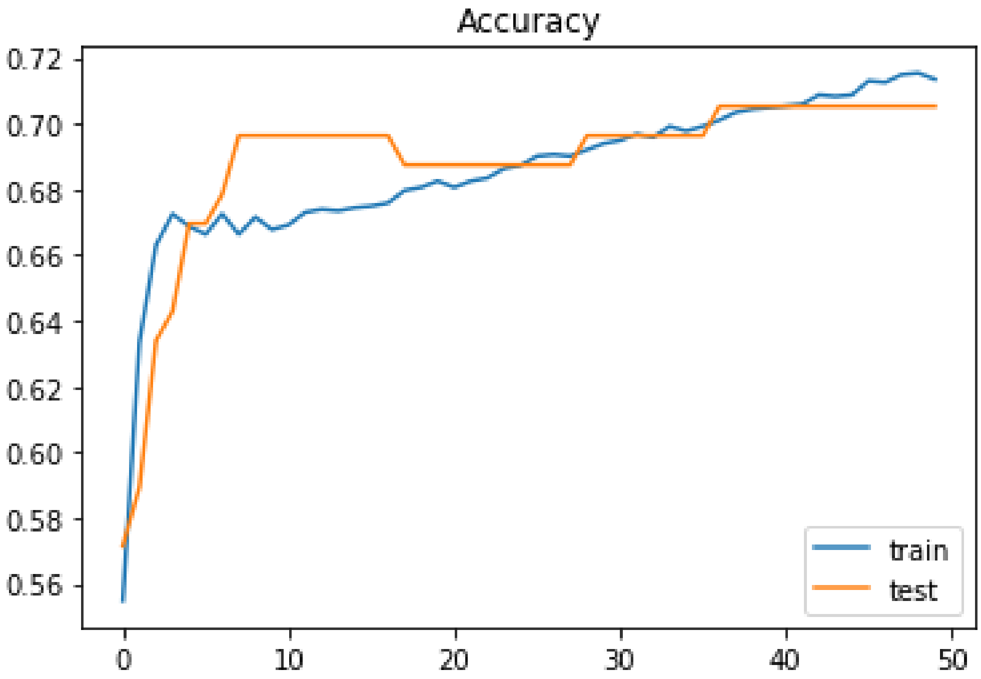
[61]:

```
# -------------------------------------------------------------
# SCRIPT CONFIGURATION
# -------------------------------------------------------------
CSV_FORMAT = CSV_FORMATS.METRICS

TRAINING_EPOCHS = 50
CLASS_WEIGHTS = {0:1, 1:2}

APPLY_FIT_VALIDATION = True
K_FOLD_VALUE = 10

EXPORT_MODEL = False

# -------------------------------------------------------------
# EXECUTION
# -------------------------------------------------------------
if CSV_FORMAT == CSV_FORMATS.METRICS:
  CSV_FILE = "../output/ml-dataset-23-03.csv"
  CSV_ROWS = ["Codebase", "Feature", "Cluster", "CLIP", "CRIP", "CROP", "CWOP",␣
 ↪"CIP", "COP", "CPIF", "CIOF", "Orchestrator"]
  #CSV_ROWS_TO_EXCLUDE = ["Codebase", "Cluster"]
  CSV_ROWS_TO_EXCLUDE = ["Cluster", "Codebase"]
  #CSV_ROWS_TO_EXCLUDE = []

elif CSV_FORMAT == CSV_FORMATS.OPERATIONS:
  CSV_FILE = "2021-03-16 23:41:19.csv"
  CSV_ROWS = ["Codebase", "Feature", "Cluster", "Entity", "Operation",␣
 ↪"Orchestrator"]
  #CSV_ROWS_TO_EXCLUDE = ["Codebase"]
  CSV_ROWS_TO_EXCLUDE = ["Cluster"]


dataset = read_dataset(CSV_FILE, CSV_ROWS, CSV_ROWS_TO_EXCLUDE)
# print(dataset.head())

dataset_features = dataset.copy()
dataset_labels = dataset_features.pop('Orchestrator')

# generate a trace_features array to make the splitting of the batches easier
trace_features = split_dataset_by_trace_features(dataset)
random.shuffle(trace_features)

# preprocessing
inputs = create_input_objects(dataset_features)
```

```
trace_preprocessing = create_preprocessing_logic(dataset, dataset_features, inputs)

number_trace_features = len(trace_features)
training_features_size = int(number_trace_features * (1-(K_FOLD_VALUE/100)))
validation_features_size = int((number_trace_features - training_features_size) / 2) if APPLY_FIT_VALIDATION else 0
testing_features_size = number_trace_features - training_features_size - validation_features_size

print(f"\n\nBatch size: {dataset_labels.size} | Number of trace features: {number_trace_features}")
print(f"Training size: {training_features_size} | Validation size: {validation_features_size} | Testing size: {testing_features_size}\n\n")


histories = []
labels = []
predictions = []
aucs = []

for iteration in range(K_FOLD_VALUE):
  (training_batch_features, training_batch_labels), (testing_batch_features, testing_batch_labels), (validation_batch_features, validation_batch_labels) = get_kfold_iteration_batches(
      iteration,
      dataset_features,
      dataset_labels,
      trace_features,
      training_features_size,
      validation_features_size,
      testing_features_size,
  )
  labels.append(testing_batch_labels)

  model = build_neural_network_model(
      body = [
        layers.Dense(9, activation="relu"),
        layers.Dense(16, activation="relu"),
        layers.Dense(2, activation="softmax")
      ],
      preprocessing_head = trace_preprocessing,
      inputs = inputs,
      loss = "sparse_categorical_crossentropy",
      optimizer = tf.optimizers.Adam(learning_rate=0.0001), # Adam or SGD
  )
  training_history = fit_neural_network(
      model = model,
      training_features = training_batch_features,
      training_labels = training_batch_labels,
      validation_features = validation_batch_features,
      validation_labels = validation_batch_labels,
      epochs = TRAINING_EPOCHS,
      shuffle = True,
      weights = CLASS_WEIGHTS,
  )
  histories.append(training_history)

  plot_training_results(training_history)

  testing_predictions = test_model(
      model = model,
      testing_features = testing_batch_features,
      testing_labels = testing_batch_labels,
      verbose = False,
  )
  predictions.append(testing_predictions)

  auc_value = plot_testing_results(testing_predictions, testing_batch_labels)
  aucs.append(auc_value)

 ␣
 ↪print("\n\n---------------------------------------------------------------------------------

sum_auc = 0.0
for auc_value in aucs:
    sum_auc += auc_value
mean_auc = sum_auc / len(aucs)
print(f"\nMean AUC: {mean_auc}\n")


if EXPORT_MODEL:
  filename = f'trace_trained_model-{datetime.now().
 ↪strftime("%d_%m_%Y_%H_%M_%S")}'
  model.save(filename)
```

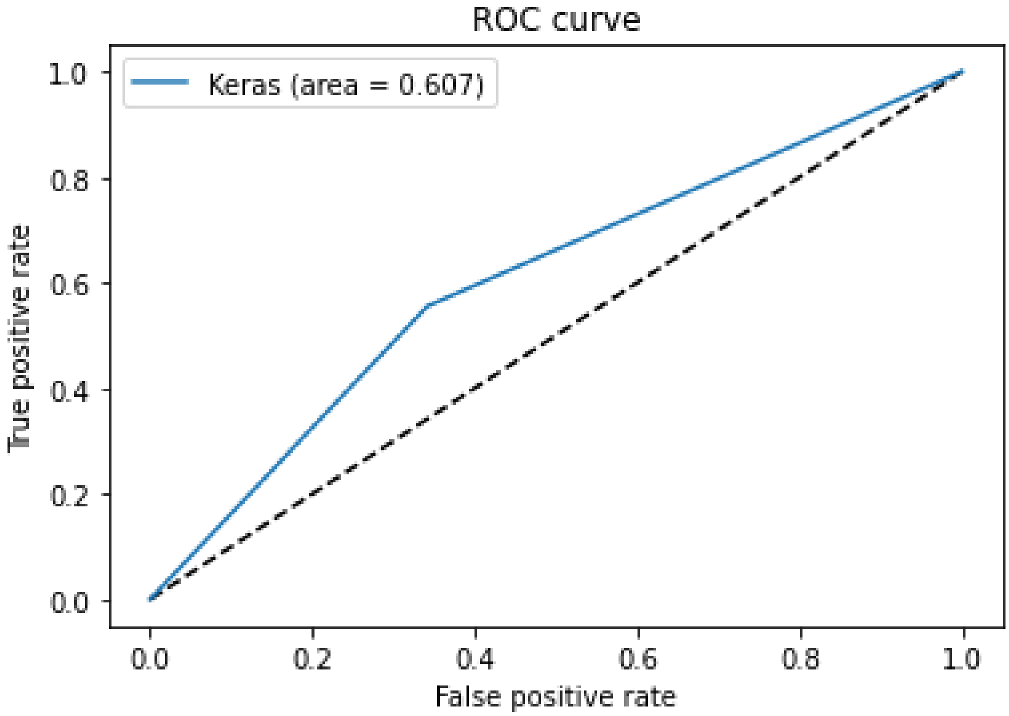
```
('Failed to import pydot. You must `pip install pydot` and install graphviz
(https://graphviz.gitlab.io/download/), ', 'for `pydotprint` to work.')
Batch size: 2325 | Number of trace features: 717
Training size: 645 | Validation size: 36 | Testing size: 36
```



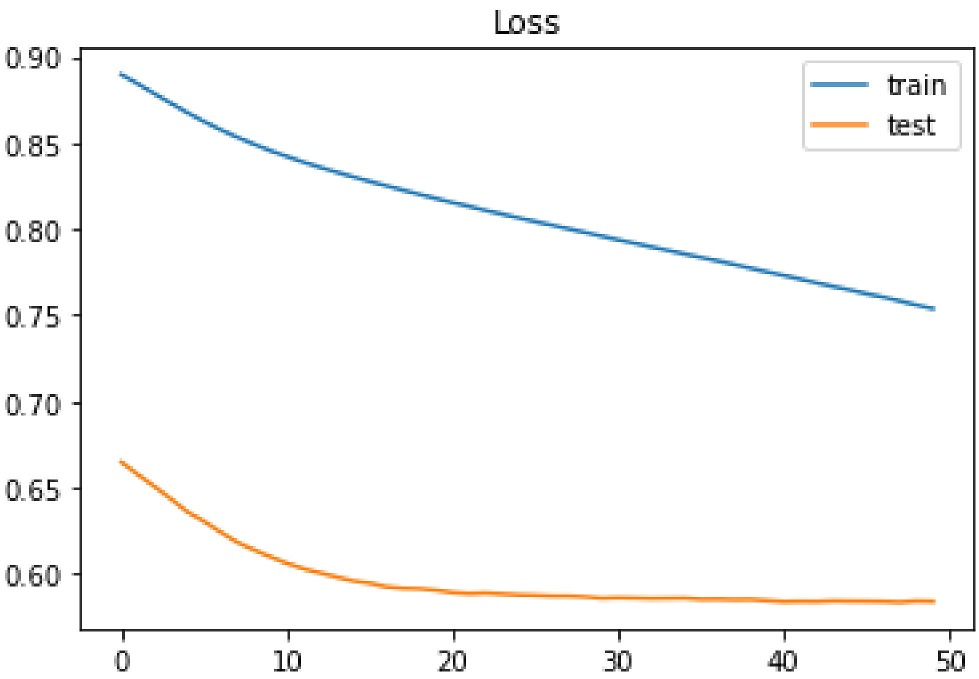


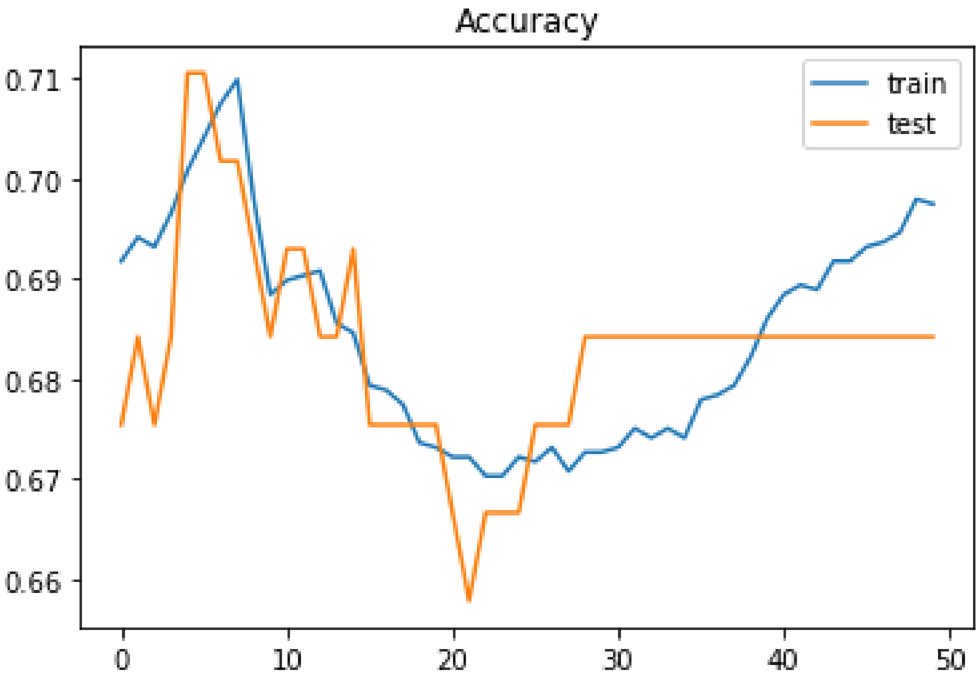
```
Training results:
```

```
Final loss: 0.7486156821250916
Final accuracy: 0.7135366797447205

Results for 115 test samples

Loss 0.5948431491851807 | Recall: 0.626086950302124
```



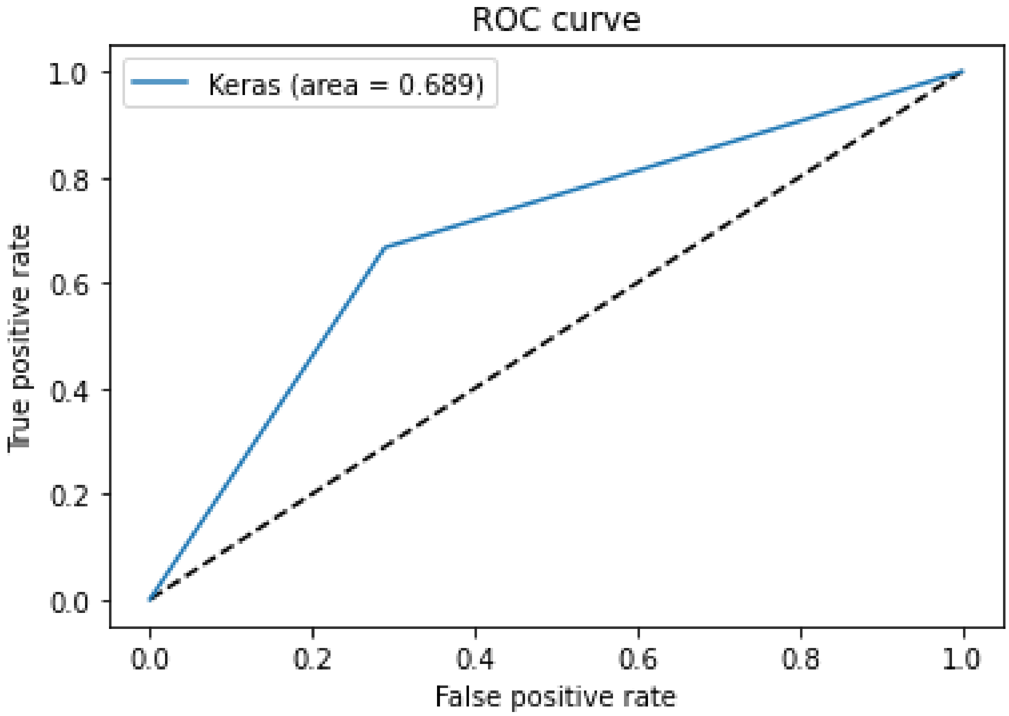
```
AUC: 0.6068917018284106


--------------------------------------------------------------------------------
------
```



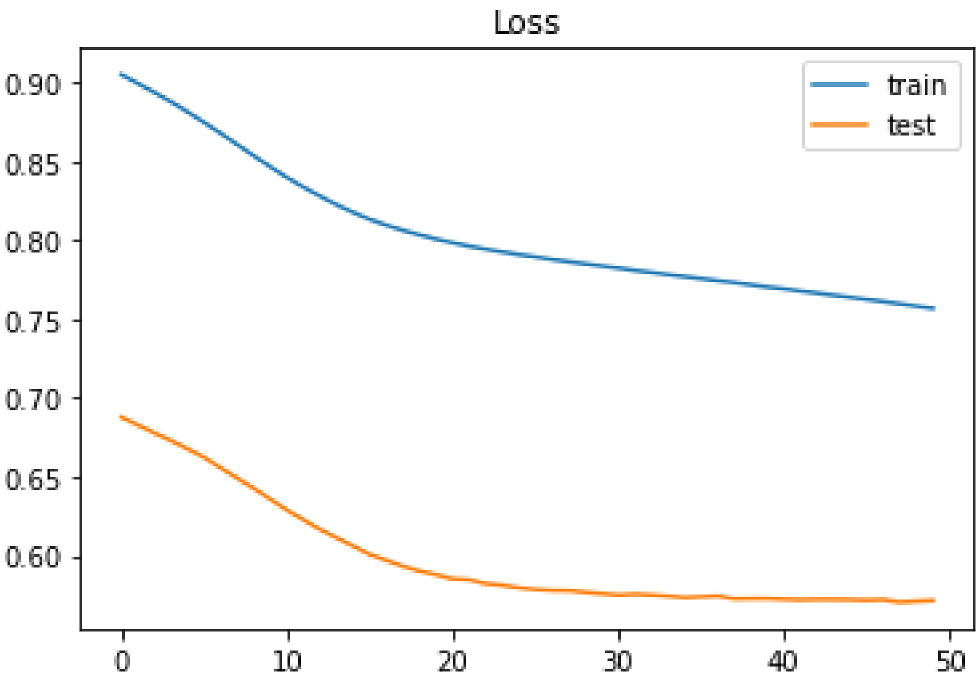


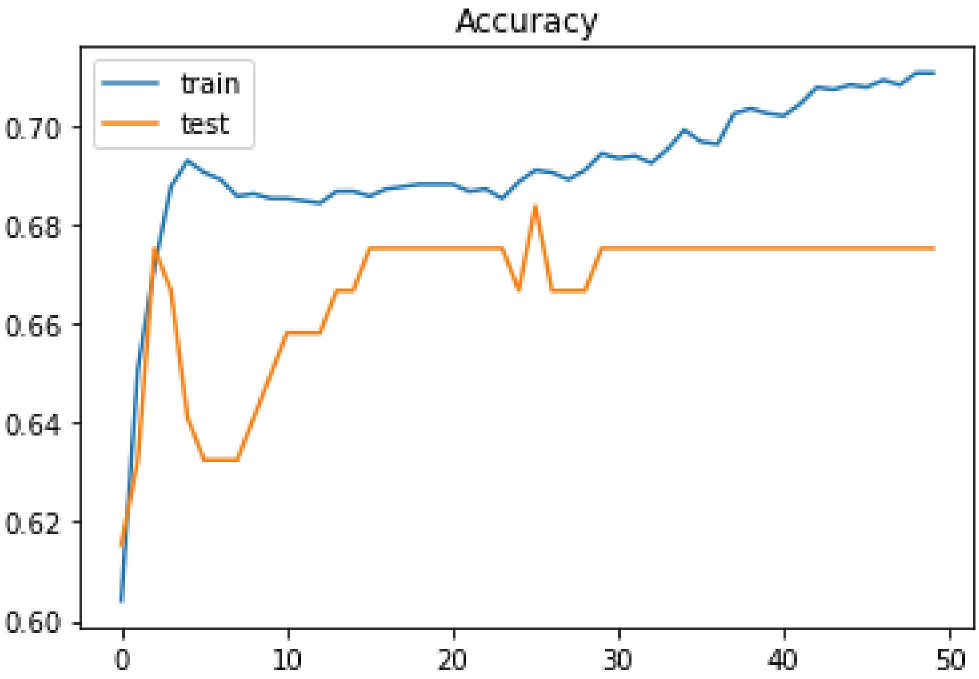
```
Training results:
```

```
Final loss: 0.753695547580719
Final accuracy: 0.697475016117096

Results for 112 test samples

Loss 0.5979799628257751 | Recall: 0.6964285969734192
```



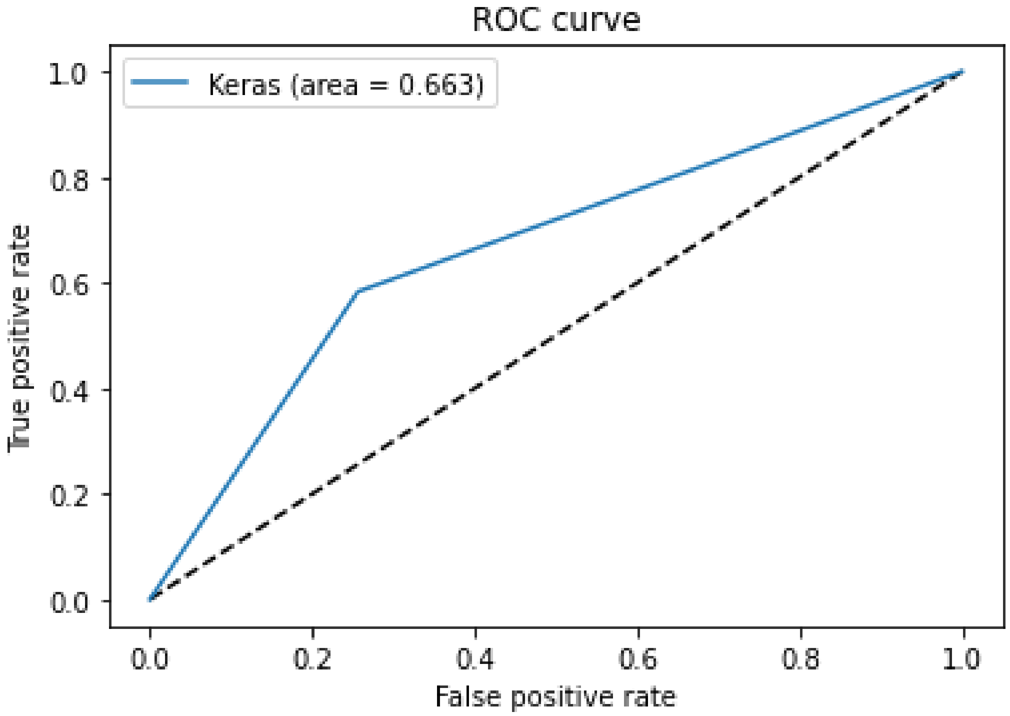
```
AUC: 0.6885964912280701


--------------------------------------------------------------------------------
------
```



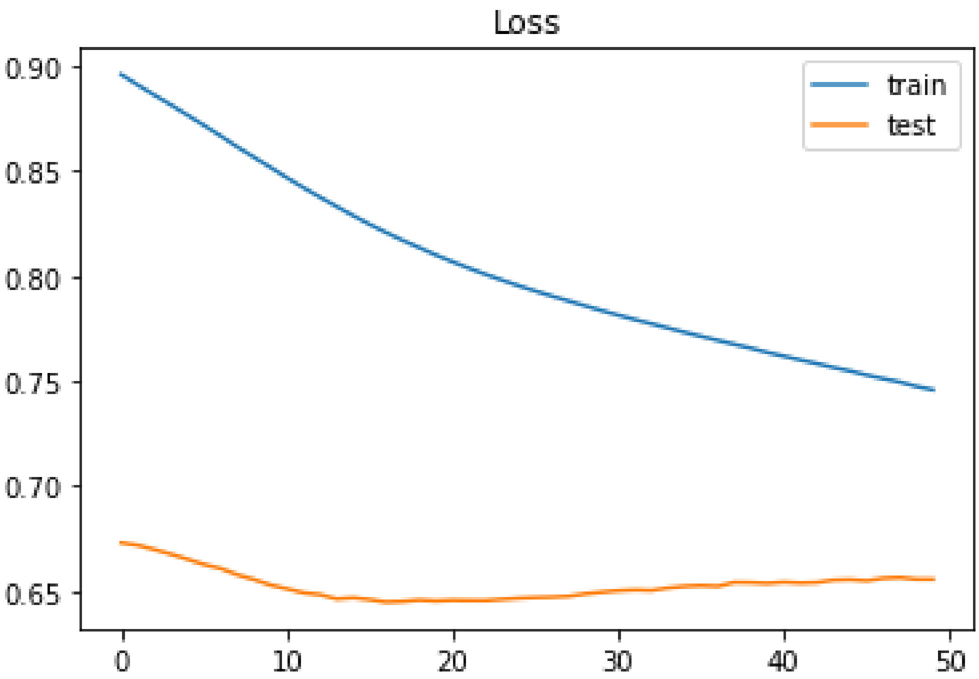


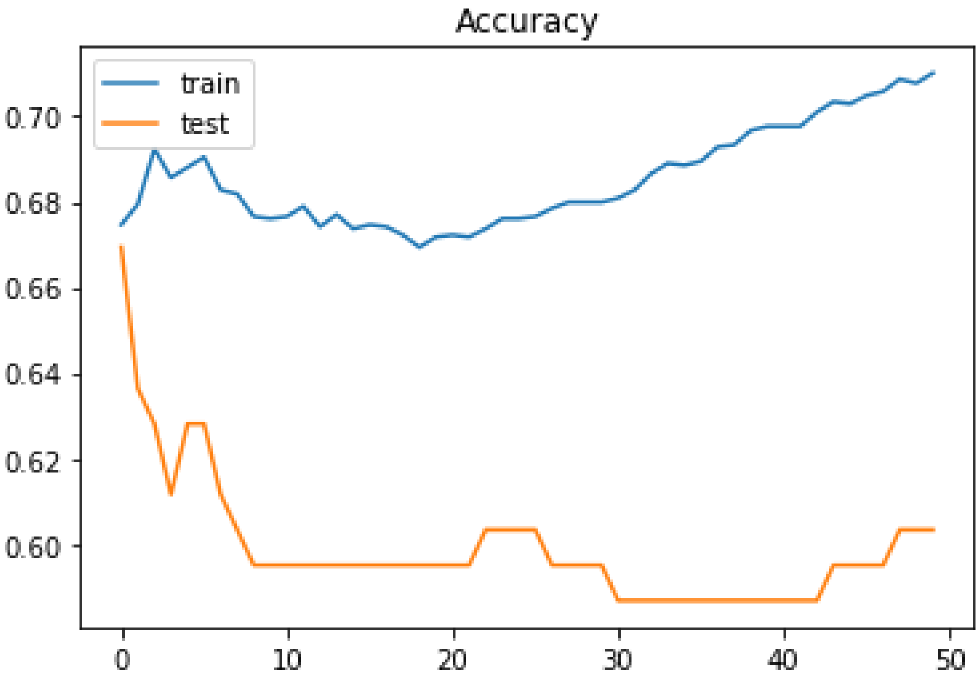
```
Training results:
```

```
Final loss: 0.7568369507789612
Final accuracy: 0.7106017470359802

Results for 114 test samples

Loss 0.5917031764984131 | Recall: 0.6929824352264404
```



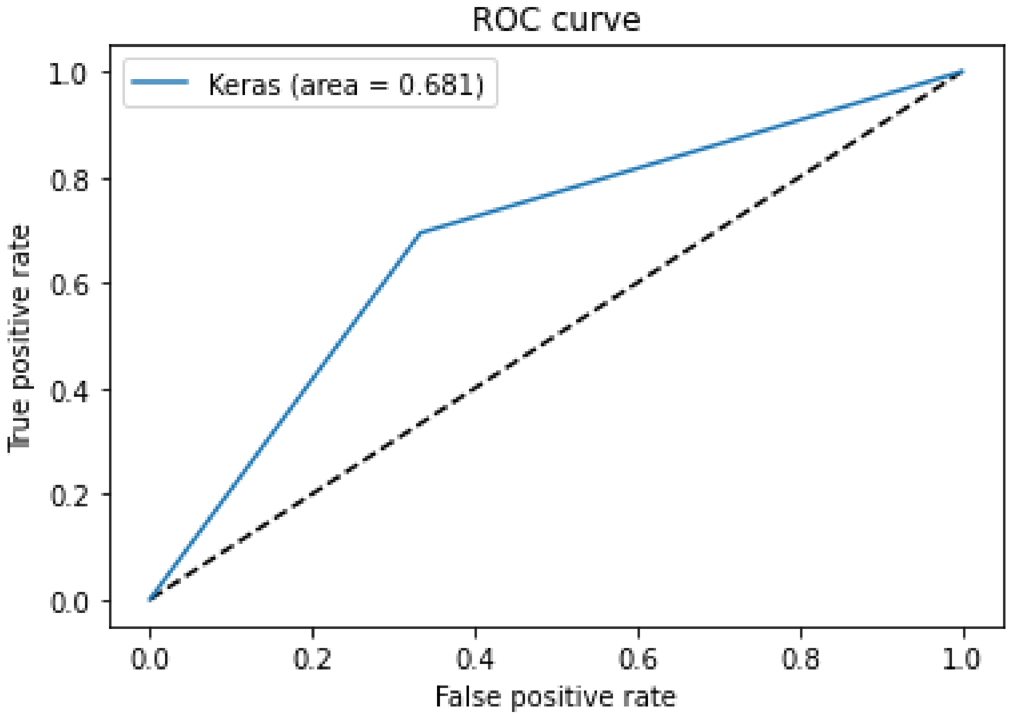
```
AUC: 0.6634615384615384


--------------------------------------------------------------------------------
------
```



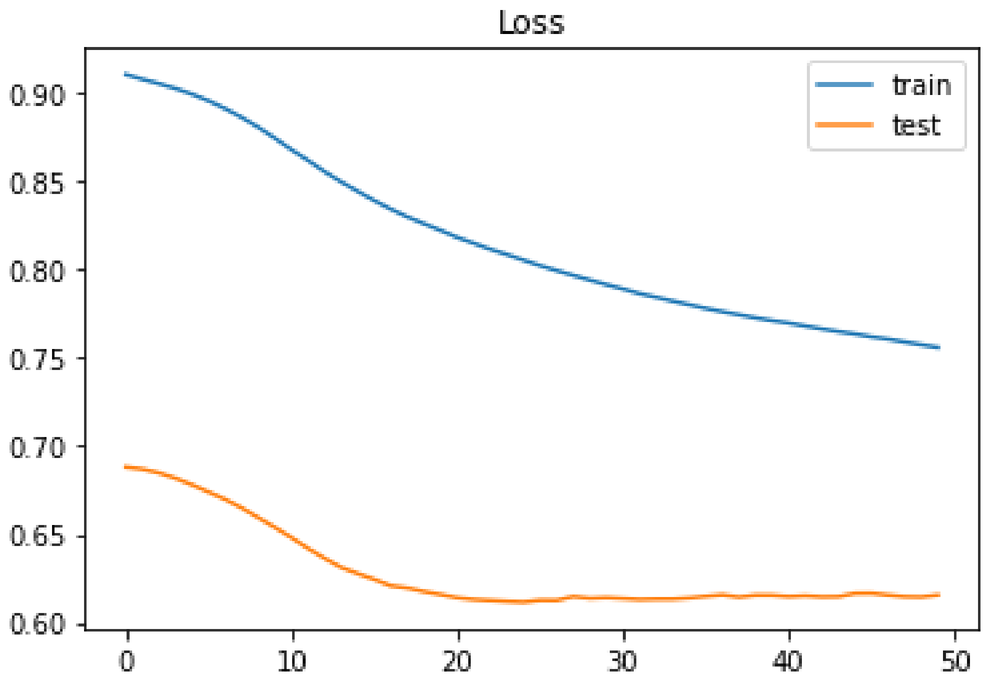


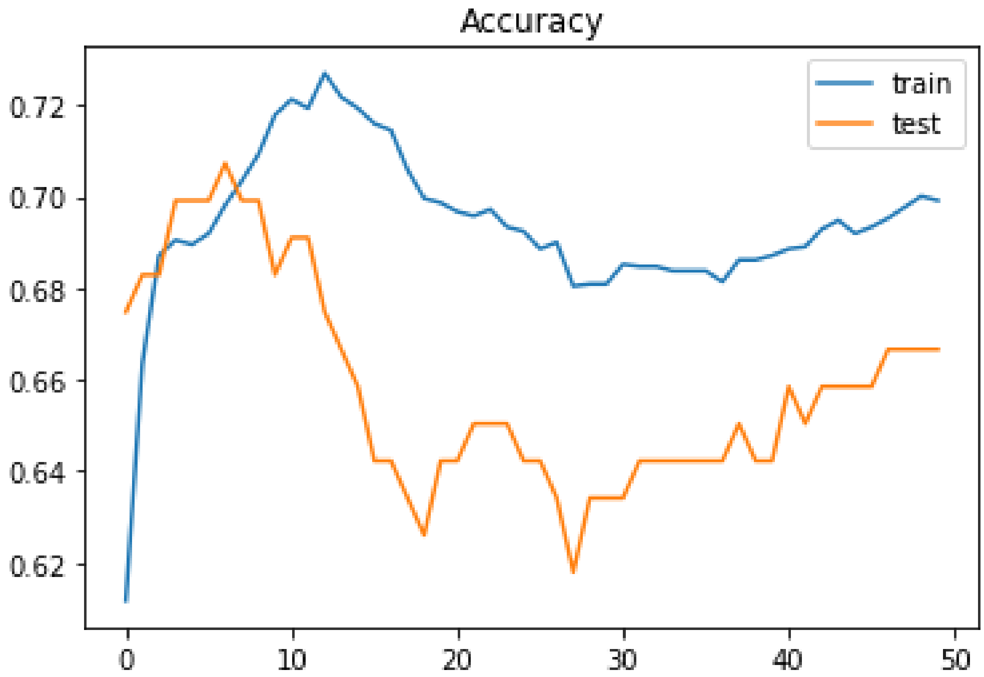
```
Training results:
```

```
Final loss: 0.7457491755485535
Final accuracy: 0.7101101875305176

Results for 117 test samples

Loss 0.5599515438079834 | Recall: 0.6752136945724487
```



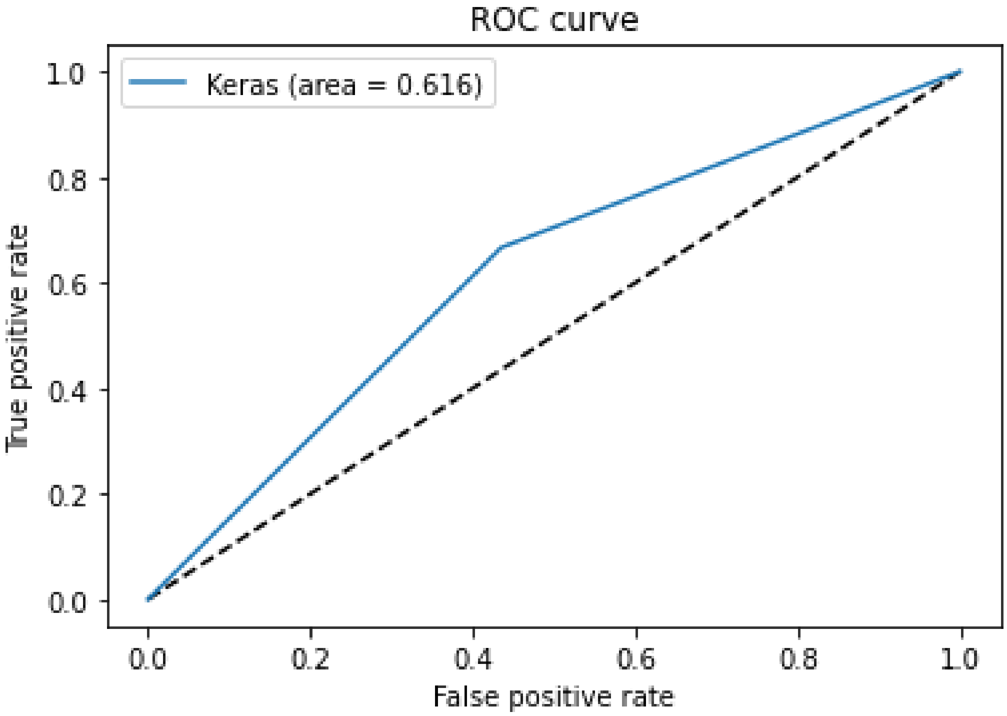
```
AUC: 0.6805555555555556


--------------------------------------------------------------------------------
------
```



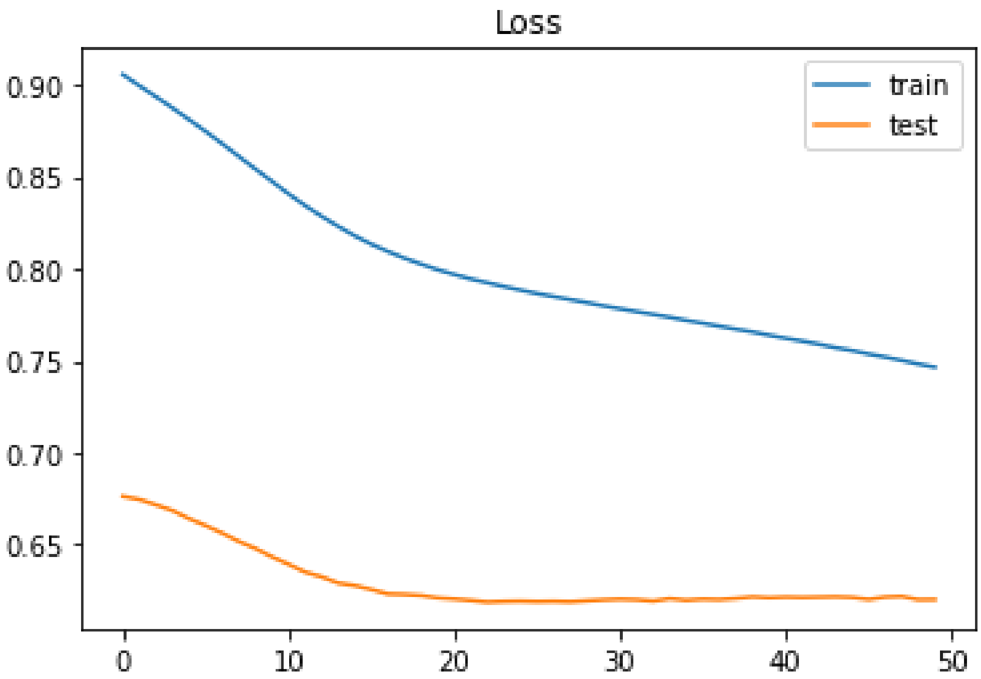


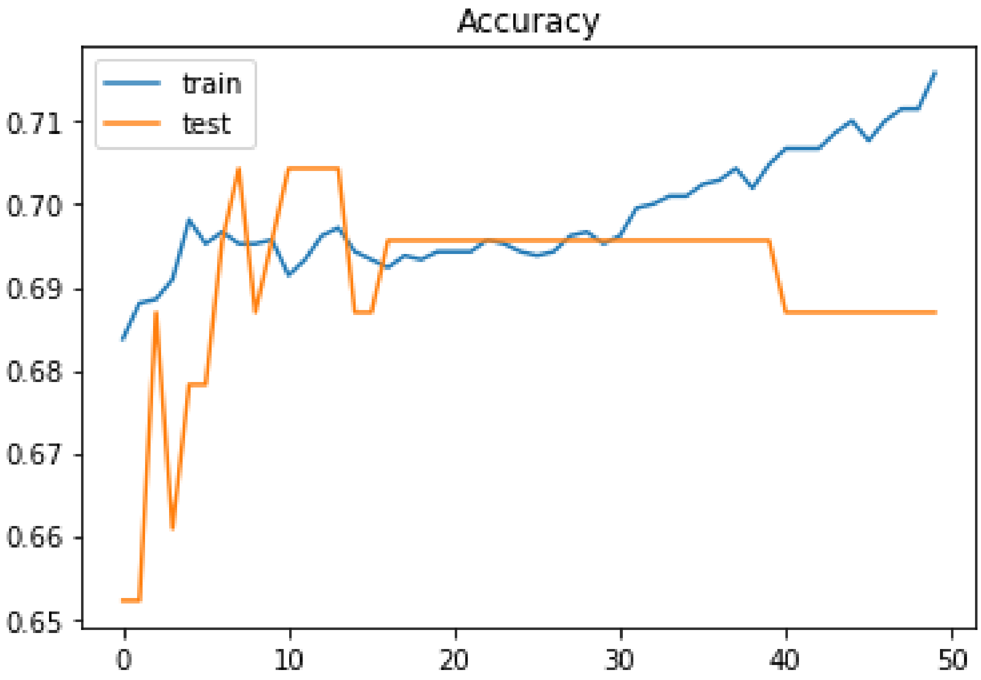
```
Training results:
```

```
Final loss: 0.7557787299156189
Final accuracy: 0.6991831064224243

Results for 121 test samples

Loss 0.6653162837028503 | Recall: 0.5950413346290588
```



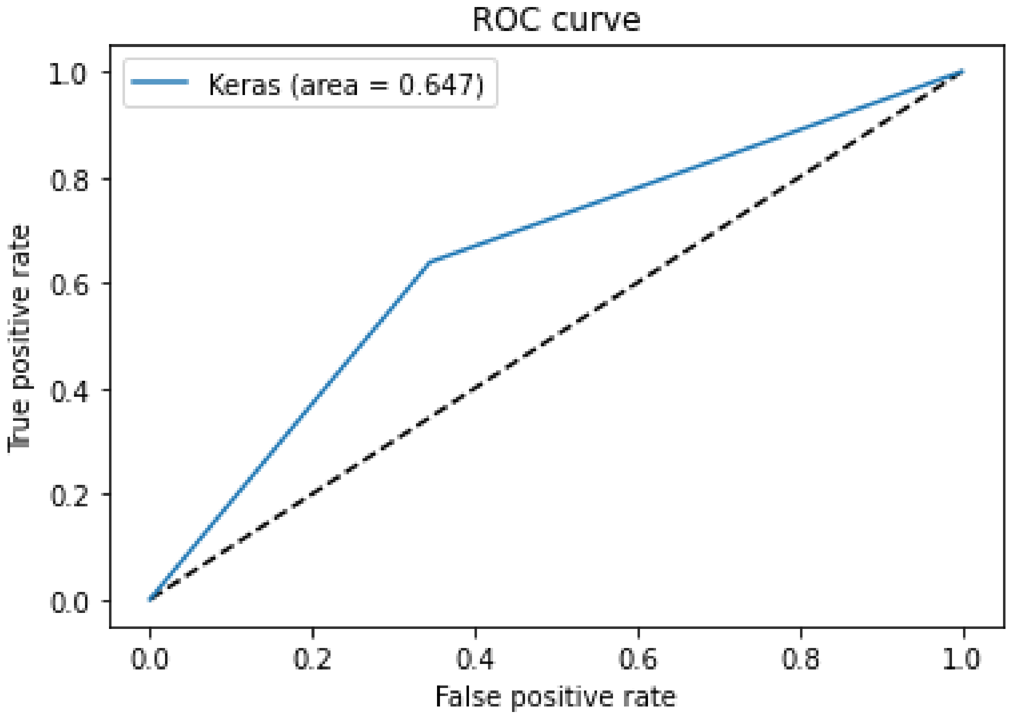
```
AUC: 0.6156862745098038


--------------------------------------------------------------------------------
------
```



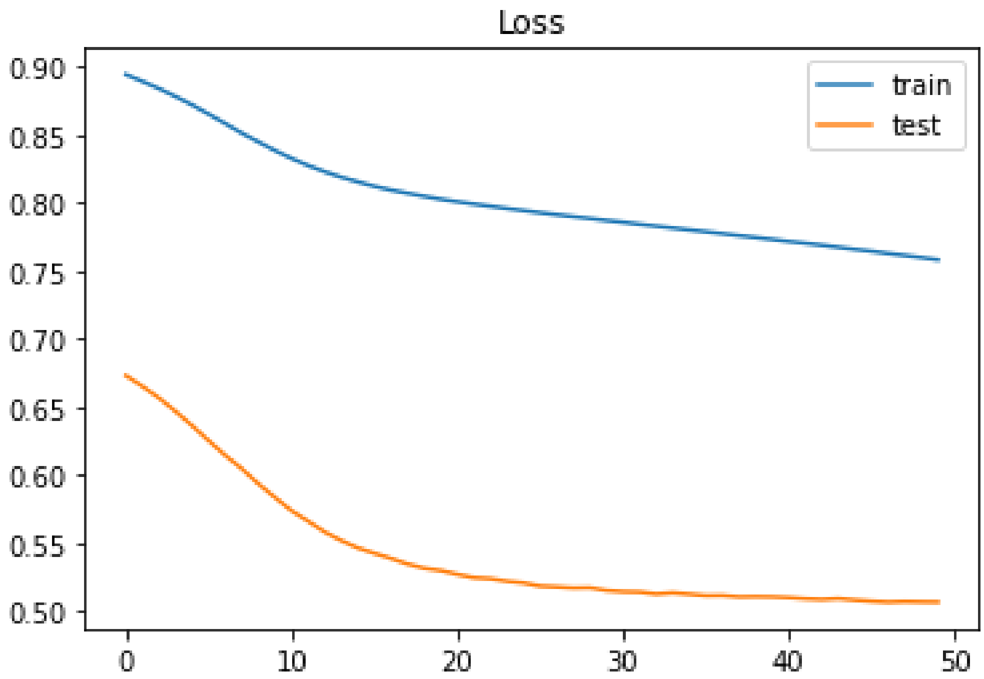


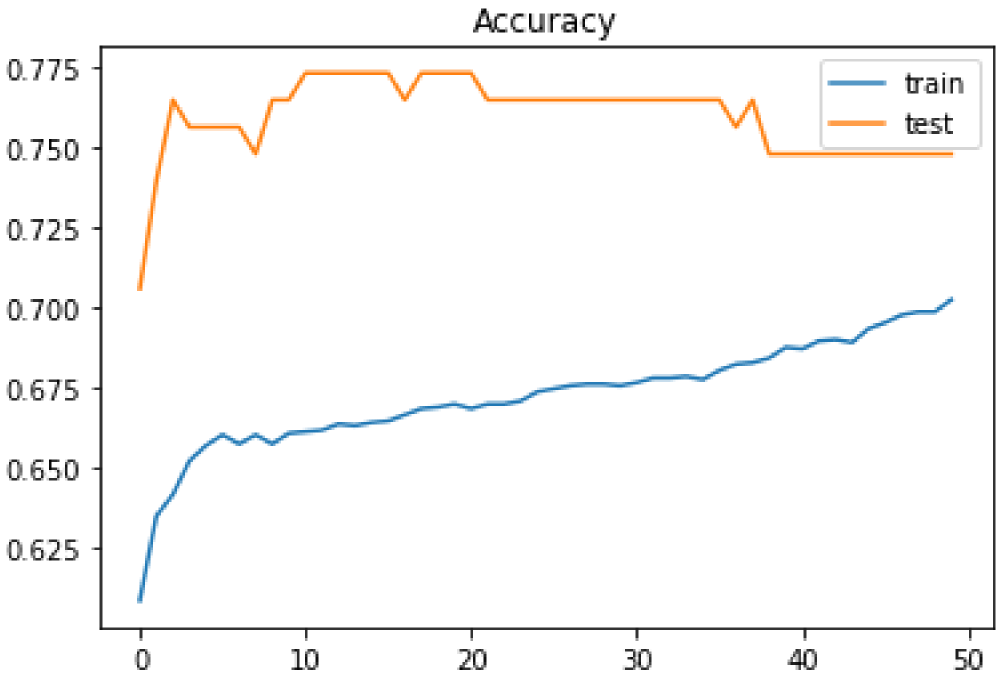
Training results:

```
Final loss: 0.7465977668762207
Final accuracy: 0.7158600687980652

Results for 123 test samples

Loss 0.6134207844734192 | Recall: 0.6504064798355103
```



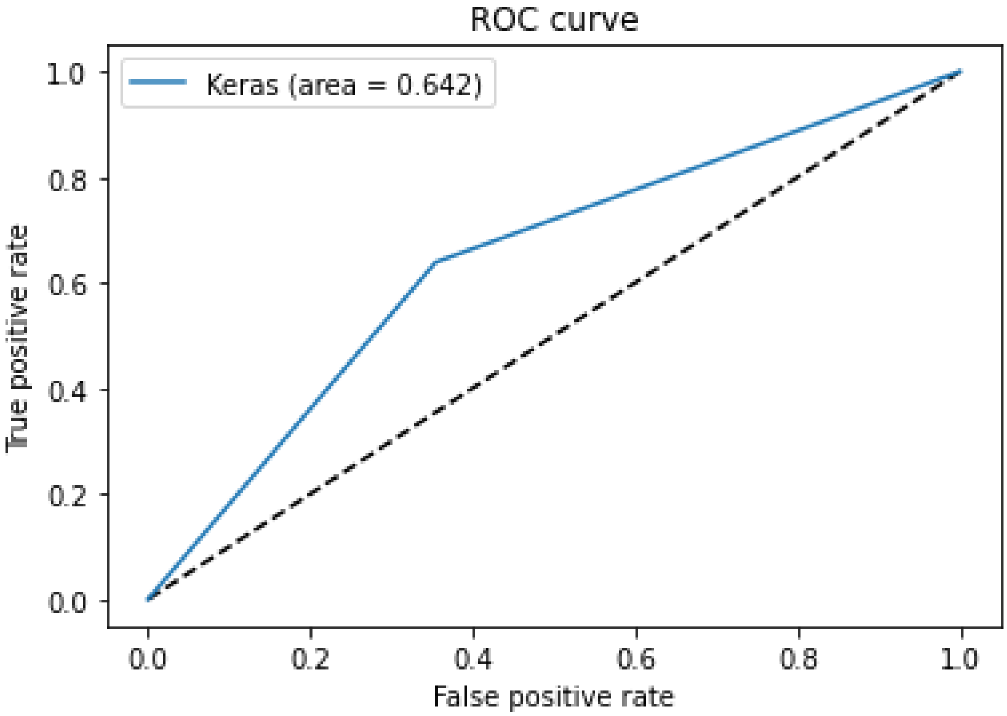
```
AUC: 0.6470306513409961


--------------------------------------------------------------------------------
------
```



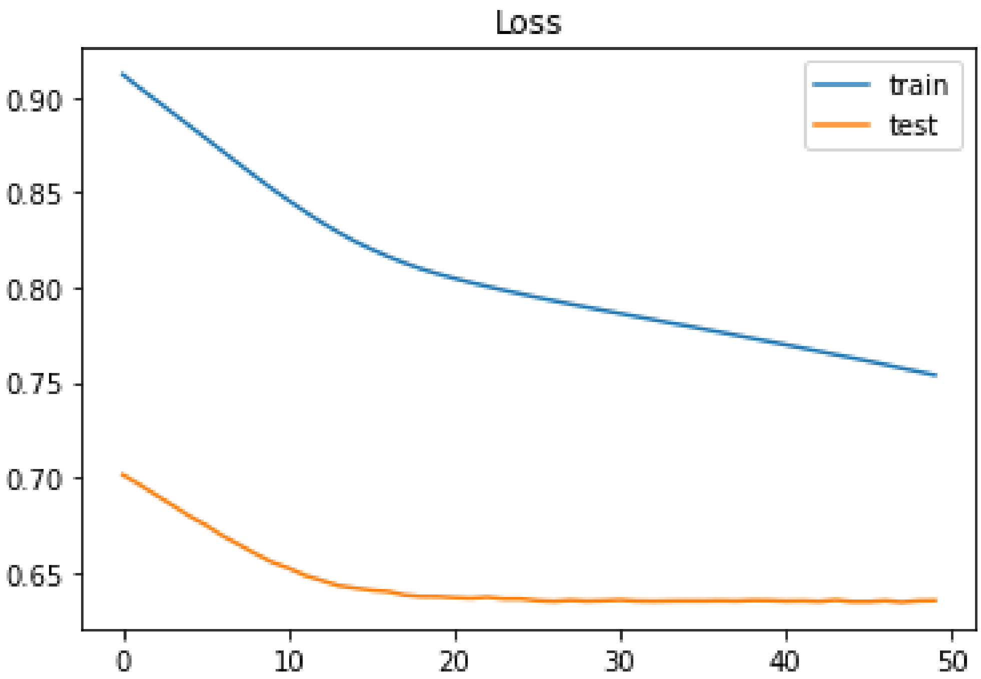


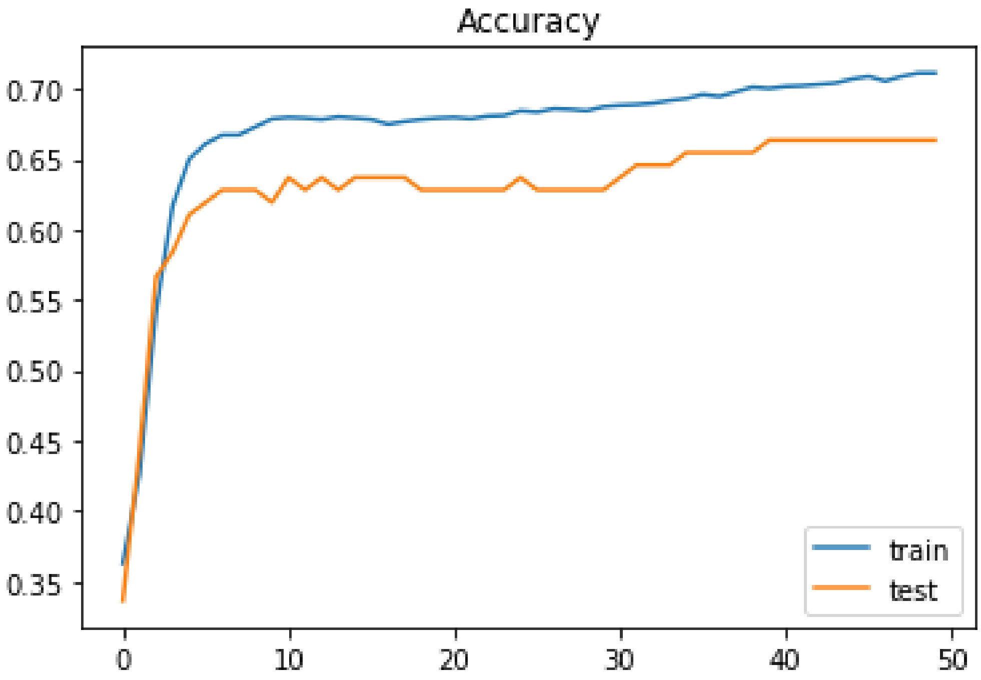
```
Training results:
```

```
Final loss: 0.7583835124969482
Final accuracy: 0.7025346755981445

Results for 115 test samples

Loss 0.6195462346076965 | Recall: 0.643478274345398
```



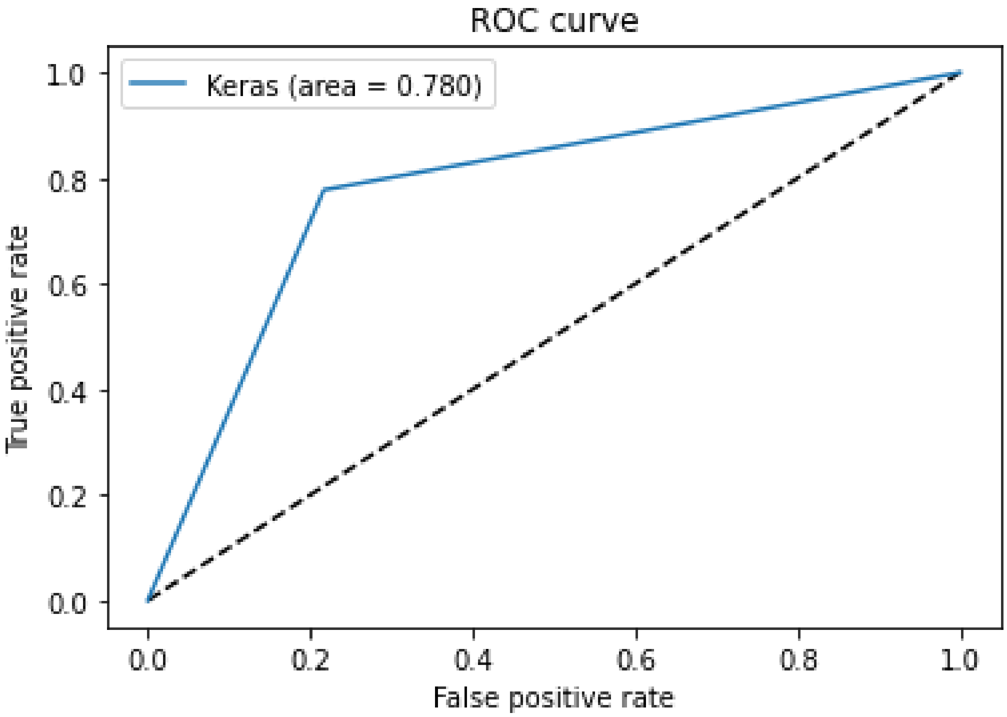
```
AUC: 0.6422292545710268


--------------------------------------------------------------------------------
------
```



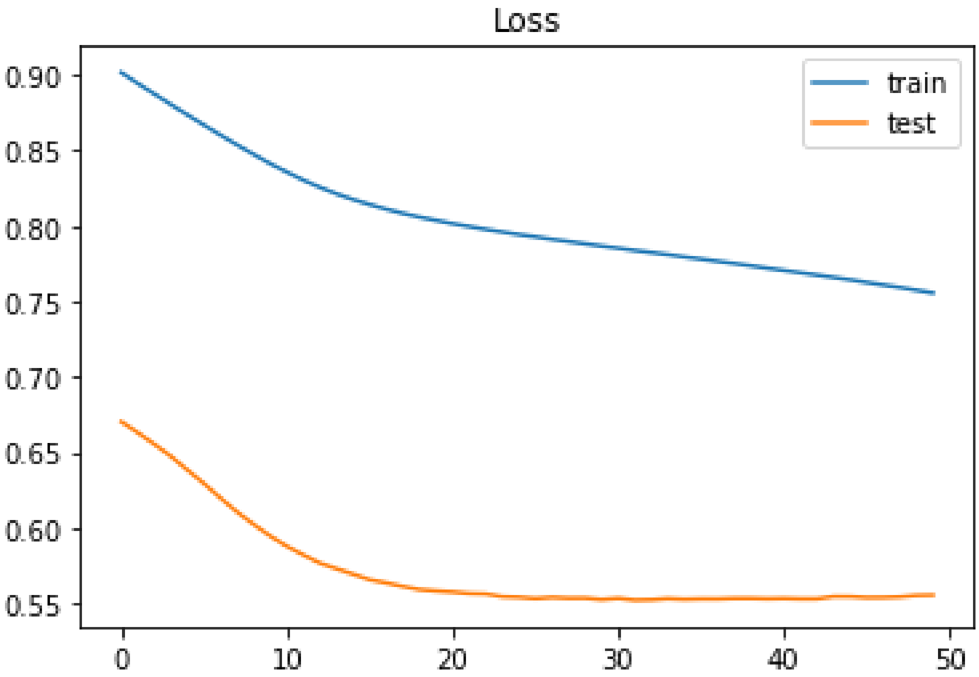


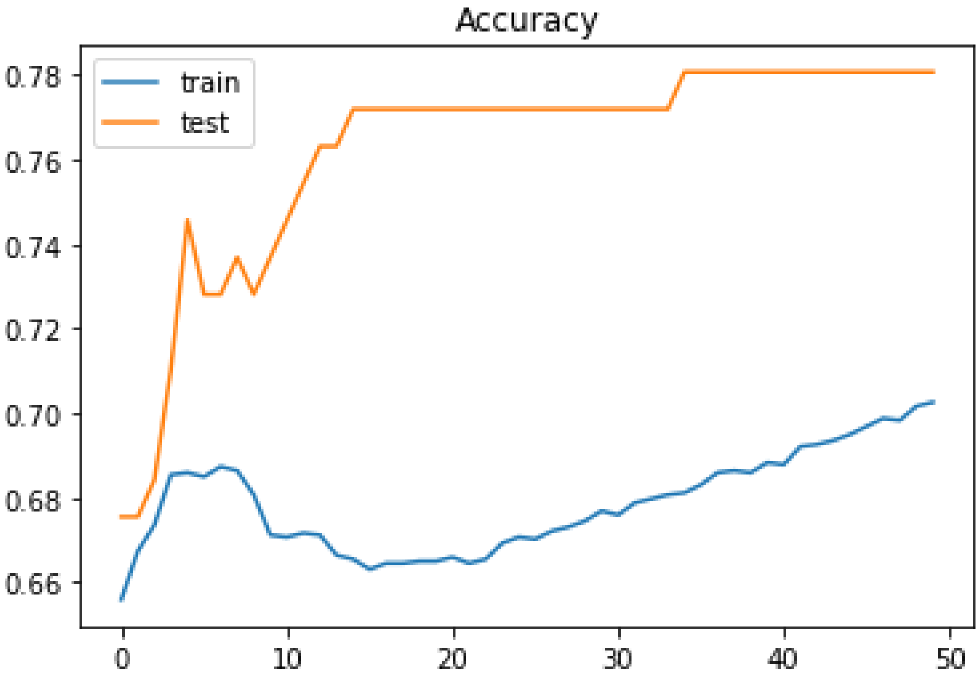
```
Training results:
```

```
Final loss: 0.7539101839065552
Final accuracy: 0.7114189863204956

Results for 119 test samples

Loss 0.5018633008003235 | Recall: 0.7815126180648804
```



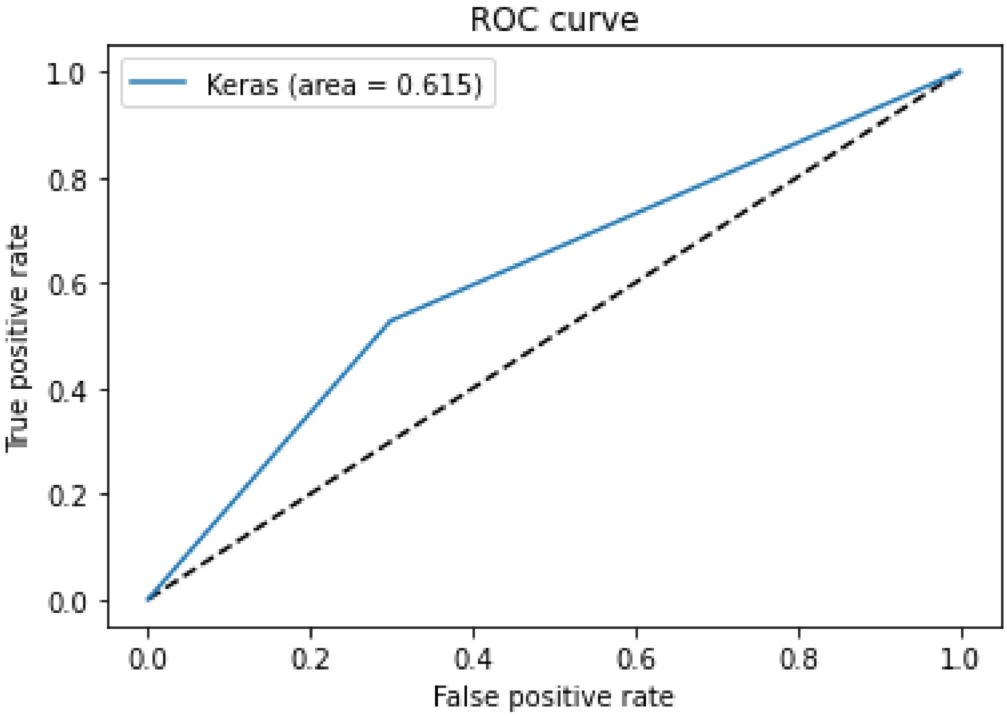
```
AUC: 0.7804551539491298


--------------------------------------------------------------------------------
------
```



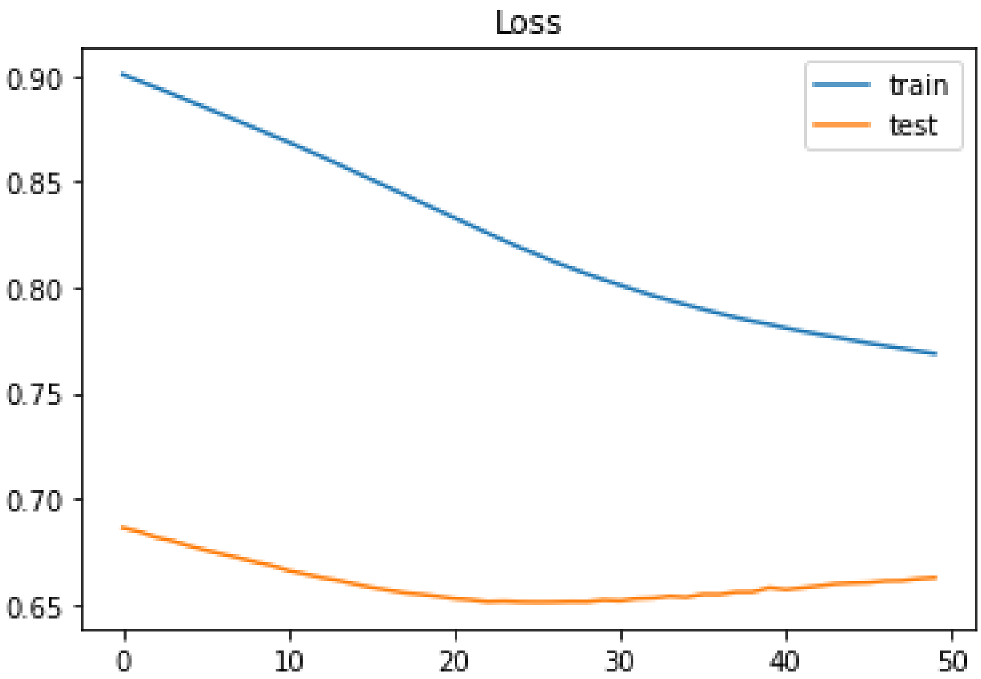


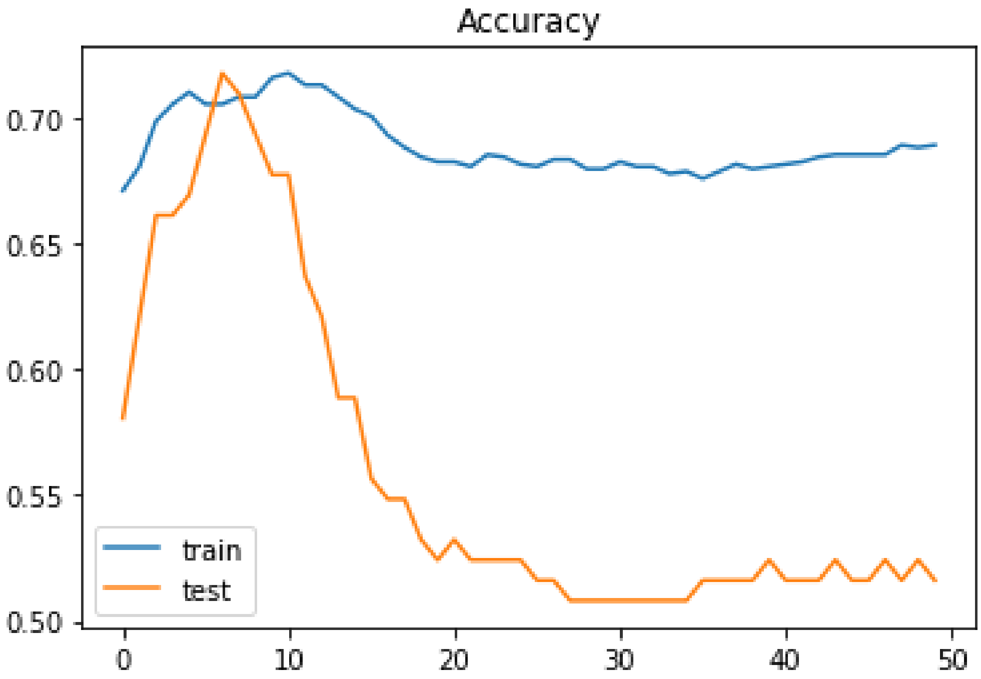
```
Training results:
```

```
Final loss: 0.7555235028266907
Final accuracy: 0.7025738954544067

Results for 113 test samples

Loss 0.6260712146759033 | Recall: 0.6460176706314087
```



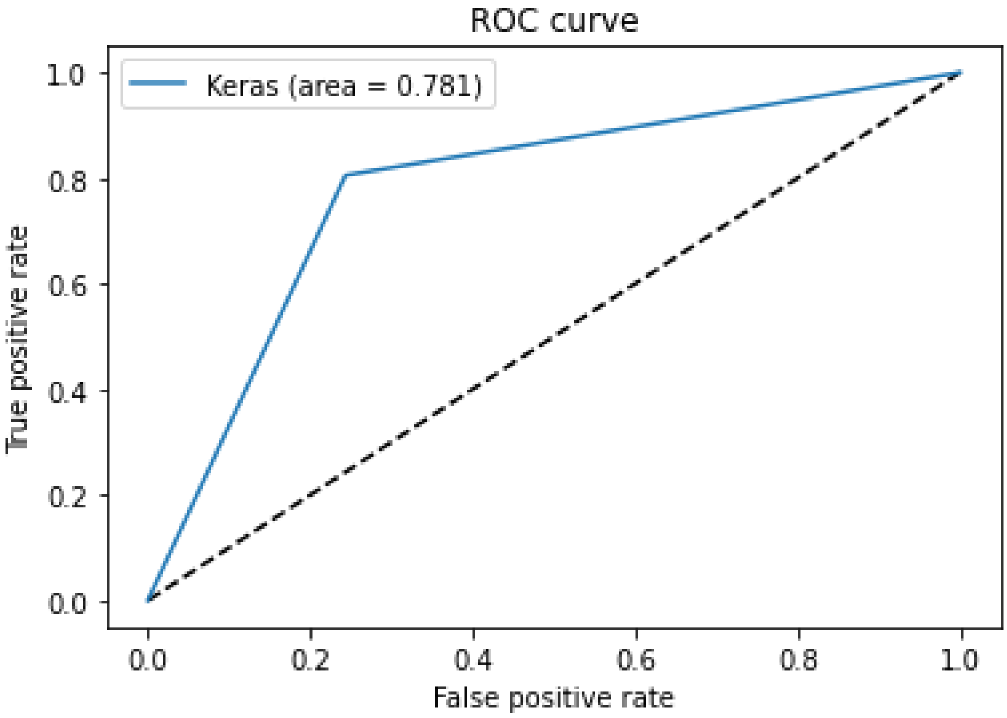
```
AUC: 0.6145382395382395


--------------------------------------------------------------------------------
------
```





```
Training results:
```

```
Final loss: 0.7686277627944946
Final accuracy: 0.689227819442749

Results for 114 test samples

Loss 0.5560402274131775 | Recall: 0.7719298005104065
```



```
AUC: 0.780982905982906


--------------------------------------------------------------------------------
------



Mean AUC: 0.6720427766965676
```