Оглавление

[**Условие задачи** 2](#_Toc87287551)

[**Разбор задачи** 3](#_Toc87287552)

[**Обучение нейронной сети** 8](#_Toc87287553)

[**Проверка нейронной сети** 10](#_Toc87287554)

[**Заключение** 12](#_Toc87287555)

[**Список литературы** 13](#_Toc87287556)

# **Условие задачи**

Имеется набор данных: изображения собак (1100 экземпляров) и кошек (1099 экземпляров). Необходимо, используя Matlab и Deep Learning Toolbox, построить модель машинного обучения, которая научится отличать изображения одного класса от другого.

Набор данных является частью коллекции картинок [Cats-vs-Dogs](https://www.kaggle.com/shaunthesheep/microsoft-catsvsdogs-dataset).

# **Разбор задачи**

При решении использован подход Transfer Learning: нейронная сеть, ранее обученная на большом наборе изображений в течение длительного времени, модифицирована для решения задачи бинарной классификации (изменены несколько последних слоёв) и дообучена в течение сравнительно небольшого времени.

Вначале для воспроизводимости фиксируем источник неопределённости для генератора псевдослучайных чисел, который используется при подготовке и обучении:

rng("default") % default seed for reproducibility

Загружаем данные и псевдослучайным образом распределяем на тренировочную и валидационную части в отношении 7 к 3:

%% load data, split train:validate as 7:3

data = imageDatastore("petImagesSelected","IncludeSubfolders", true, "LabelSource","foldernames");

data.shuffle();

[dataTrain, dataValidate] = splitEachLabel(data, .7);

%% explore the dataset

namedDatasets = {

"train", dataTrain;

"validate", dataValidate};

for i = 1:size(namedDatasets,1)

title = namedDatasets{i,1};

dataSubset = namedDatasets{i,2};

disp(title + ": sample size: " + numel(dataSubset.Files));

disp(title + ": labels distribution: ")

disp(dataSubset.countEachLabel())

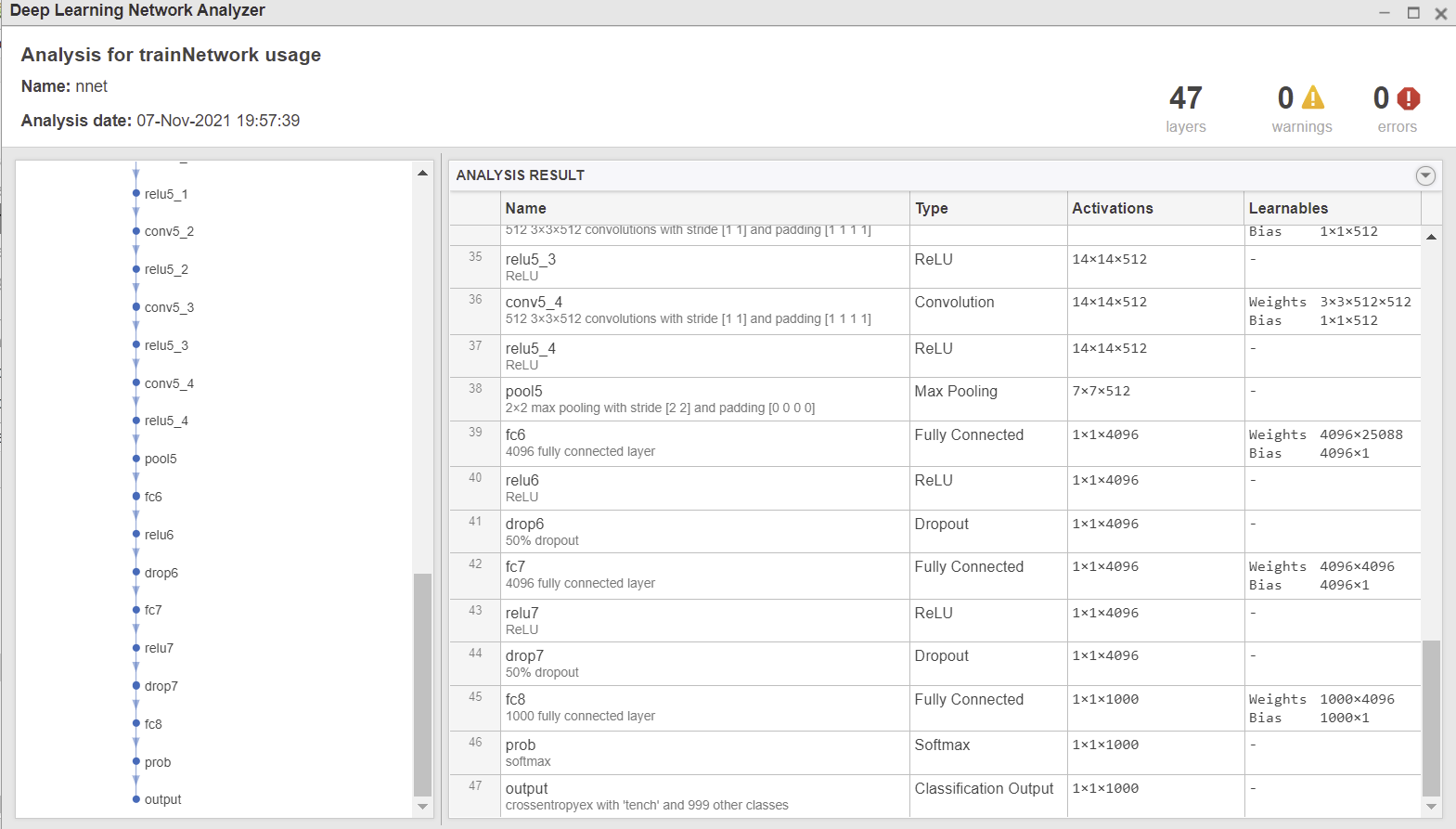
end

В качестве базовой модели будем использовать нейронную сеть VGG-19 [[1]](#_Список_литературы), хорошо зарекомендовавшую себя в задачах классификации изображений:

%% look at the pre-trained convolutional neural network

nnet = vgg19;

analyzeNetwork(nnet);



Заменим полносвязный слой, идущий перед softmax-слоем, на новый, который будет соответствовать нашему числу классов (2). Выходной слой нейронной сети также заменим на новый. В первых десяти слоях нейросети положим значения параметров: “weightLearnRateFactor” и “biasLearnRateFactor” равными нулю, тем самым запретив изменять в них веса при тренировке. Перестроим связи в графе нейросети.

%% set up the new network layers (all pre-trained except for two)

nClasses = numel(categories(data.Labels));

lgraph = layerGraph(nnet);

lgraph = replaceLayer(lgraph,lgraph.Layers(end-2).Name, ...

fullyConnectedLayer(nClasses, ...

"Name", "fine\_tune\_fc", ...

"WeightLearnRateFactor", 10, ...

"BiasLearnRateFactor", 10));

lgraph = replaceLayer(lgraph,lgraph.Layers(end).Name, ...

classificationLayer("Name", "fine\_tune\_output"));

%% update layers properties

layers = lgraph.Layers;

connections = lgraph.Connections;

for ii = 1:10

props = properties(layers(ii));

for p = 1:numel(props)

propName = props{p};

if ~isempty(regexp(propName, 'LearnRateFactor$', 'once'))

layers(ii).(propName) = 0;

end

end

end

lgraph = layerGraph();

for i = 1:numel(layers)

lgraph = addLayers(lgraph,layers(i));

end

for c = 1:size(connections,1)

lgraph = connectLayers(lgraph,connections.Source{c},connections.Destination{c});

end

Вывод в консоли:

train: sample size: 1539

train: labels distribution:

**Label**  **Count**

**\_\_\_\_\_**  **\_\_\_\_\_**

cats 769

dogs 770

validate: sample size: 660

validate: labels distribution:

**Label**  **Count**

**\_\_\_\_\_**  **\_\_\_\_\_**

cats 330

dogs 330

Зададим правила аугментации изображений при тренировке: их можно отражать вдоль вертикальной оси и поворачивать на угол

%% set up data augmenter

augmenter = imageDataAugmenter( ...

"RandXReflection", true, ...

"RandRotation", [-45, 45]);

inputDims = layers(1).InputSize(1:2);

augTrain = augmentedImageDatastore(inputDims,dataTrain, 'DataAugmentation',augmenter, "ColorPreprocessing","gray2rgb");

augValidate = augmentedImageDatastore(inputDims,dataValidate, "ColorPreprocessing","gray2rgb");

Укажем параметры оптимизации стохастическим градиентным спуском: параметр скорости обучения – 0.001, максимальное число эпох – 16, размер пакета картинок – 22 в каждой итерации, данные перемешиваются после каждой завершённой эпохи, валидация – раз в 10 итераций спуска.

%% set up optimization parameters

miniBatchSize = 22;

options = trainingOptions("sgdm", ...

"ExecutionEnvironment","auto",...

"InitialLearnRate",0.001,...

"MaxEpochs", 16,...

"MiniBatchSize", 22, ...

"Shuffle","every-epoch", ...

"ValidationData",augValidate, ...

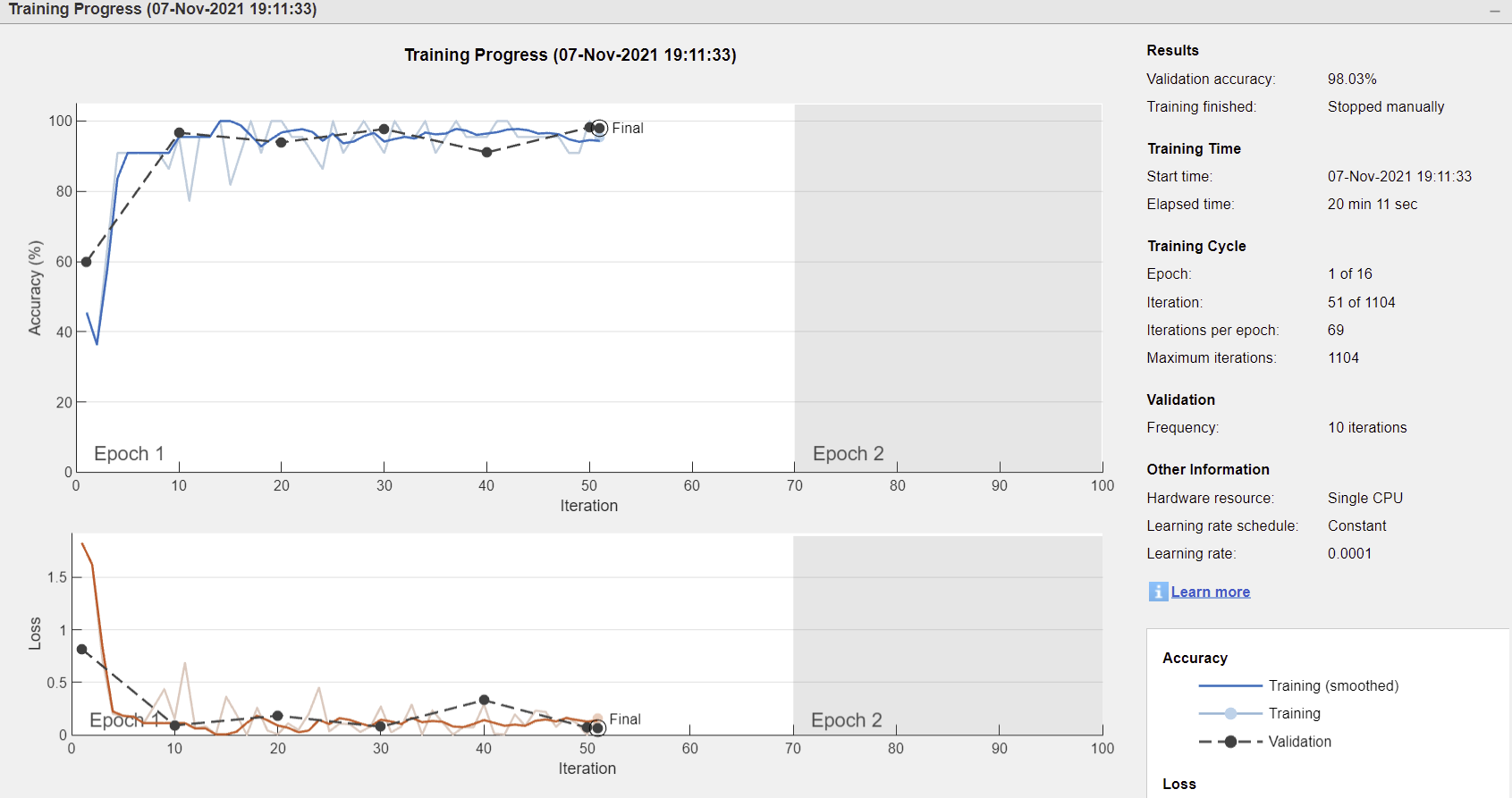
"ValidationFrequency", 10, ...

"Verbose",false, ...

"Plots","training-progress");

# **Обучение нейронной сети**

Тренируем. Тренировка занимает много времени, поэтому останавливаем её, когда точность распознавания на валидационном наборе достигает приемлемого уровня:



Посмотрим на результаты классификации валидационного набора обученной нейронной сетью:

%% evaluate the net

[yPredicted,scores] = classify(nnetTransfer,augValidate);

yGroundTruth = dataValidate.Labels;

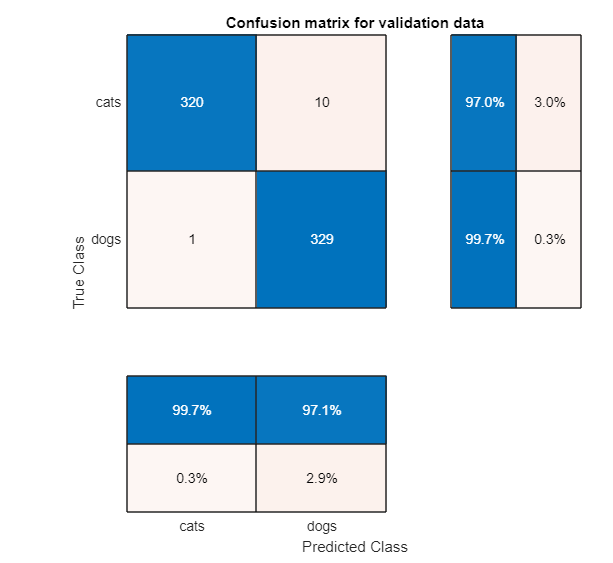
figure('Units','normalized','Position',[0.2 0.2 0.4 0.4]);

cm = confusionchart(yGroundTruth,yPredicted);

cm.Title = 'Confusion matrix for validation data';

cm.ColumnSummary = 'column-normalized';

cm.RowSummary = 'row-normalized';



Видно, что 10 кошек ошибочно отнесены в класс собак и 1 собака ошибочно распознана как кошки. Остальные экземпляры из валидационного набора определены верно.

# **Проверка нейронной сети**

Теперь проверим, как нейронная сеть справляется с изображениями кошек и собак, взятых из интернета.

Возьмем два изображения из интернета:

Изображение выглядит как текст, кот, внутренний, млекопитающее

Автоматически созданное описание

Изображение выглядит как текст, собака, внешний, земля

Автоматически созданное описание

Теперь в программу добавим новый блок:

%%

myData = imageDatastore("myPictures", "IncludeSubfolders", true, "LabelSource","foldernames");

myAugmented = augmentedImageDatastore(inputDims,myData);

classes = classify(nnetTransfer, myAugmented);

for i = 1:numel(classes)

disp(myAugmented.Files(i) + ": " + string(classes(i)));

end

Тестируем:



Можем заметить, что наша нейронная сеть верно определила классы животных.

# **Заключение**

В работе проведены построения классификационной модели, которая научилась отличать кошек от собак по изображениям. Использована техника transfer learning, благодаря которой с помощью предобученной свёрточной нейронной сети VGG-19 за сравнительно небольшой промежуток времени получена модель, отличающая котов от собак с точностью выше 0.98 на валидационном наборе.

# **Список литературы**

1. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015. URL: <https://arxiv.org/abs/1409.1556>