ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
**«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

**Высшая школа программной инженерии**

# ПРАКТИЧЕСКАЯ РАБОТА №1

по дисциплине «Проектирование интеллектуальных систем управления»

Студент А. М. Потапова
гр. 3530202/90202

Преподаватель Bahrami AmirHossein

Санкт-Петербург
2022 г

# Введение

Мною был выбран пример «Обучение сети глубокого обучения классифицировать новые изображения». В этом примере показано, как использовать трансферное обучение для переобучения сверточной нейронной сети для классификации нового набора изображений.

Предварительно сети классификации изображений были обучены на более чем миллионе изображений и могут классифицировать изображения по 1000 категориям объектов. Сеть принимает изображение в качестве входных данных, а затем выводит метку объекта на изображении вместе с вероятностями для каждой из категорий объектов.

Трансферное обучение обычно используется в приложениях глубокого обучения. Вы можете взять предварительно обученную сеть и использовать ее в качестве отправной точки для изучения новой задачи. Точная настройка сети с трансферным обучением обычно намного быстрее и проще, чем обучение сети с нуля со случайно инициализированными весами. Вы можете быстро перенести изученные функции в новую задачу, используя меньшее количество обучающих изображений.

## Ход работы

### 1) Загрузка данных.

Разархивация и загрузка новых изображений в качестве хранилища данных. Этот набор данных содержит всего 75 изображений.

```
unzip('MerchData.zip');

imds = imageDatastore('MerchData', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');
```

Разделение данных на обучающие и проверочные наборы (80% изображений для обучения и 20% для проверки. Параметры были изменены.

```
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.8);
```

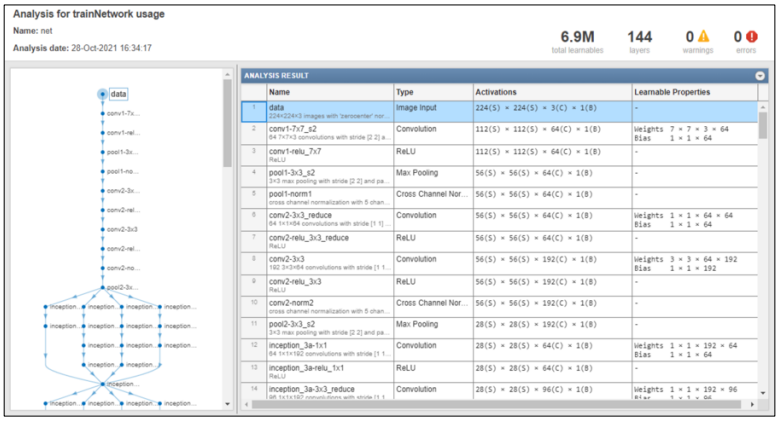
### 2) Загрузка предварительно обученной сети.

Загрузим предварительно обученную сеть *squeezend*.

```
net = squeezenet;
```

Используем analyzeNetwork для отображения интерактивной визуализации сетевой архитектуры и подробной информации о сетевых слоях.

```
analyzeNetwork(net)
```



Первым элементом Layers свойства сети является входной слой изображения. Для сети *squeezend* этот слой требует входных изображений размером 227 на 227 на 3, где 3 — количество цветовых каналов.

```
net.Layers(1)
```

```
ans =
  ImageInputLayer with properties:

                      Name: 'data'
                 InputSize: [227 227 3]
       SplitComplexInputs: 0

   Hyperparameters
          DataAugmentation: 'none'
             Normalization: 'zerocenter'
    NormalizationDimension: 'auto'
                      Mean: [1×1×3 single]
```

```
inputSize = net.Layers(1).InputSize;
```

### 3) Замена последних слоев

Сверточные слои сети извлекают признаки изображения, которые последний обучаемый слой и последний слой классификации используют для классификации входного изображения. Эти два уровня 'conv10' и 'ClassificationLayer_predictions' в *squeezend* содержат информацию о том, как объединить функции, которые извлекает сеть, в вероятности классов, значение потерь и прогнозируемые метки. Чтобы переобучить предварительно обученную сеть для классификации новых изображений, замените эти два слоя новыми слоями, адаптированными к новому набору данных.

```
lgraph = layerGraph(net);
```

Поиск двух слоев для замены. Использована вспомогательная функция findLayersToReplace для автоматического поиска этих слоев.

```
[learnableLayer,classLayer] = findLayersToReplace(lgraph);
[learnableLayer,classLayer]
```

```
ans =
  1×2 Layer array with layers:

     1   'conv10'                          2-D Convolution         1000 1×1×512 convolutions with stride [1  1] and
     2   'ClassificationLayer_predictions'   Classification Output   crossentropyex with 'tench' and 999 other classes
```

В некоторых сетях, таких как SqueezeNet, последний обучаемый слой вместо этого представляет собой сверточный слой 1 на 1. Заменим его на новый сверточный слой с количеством фильтров, равным количеству классов. Чтобы учиться на новом слое быстрее, чем на переданных слоях, увеличим коэффициенты скорости обучения слоя.

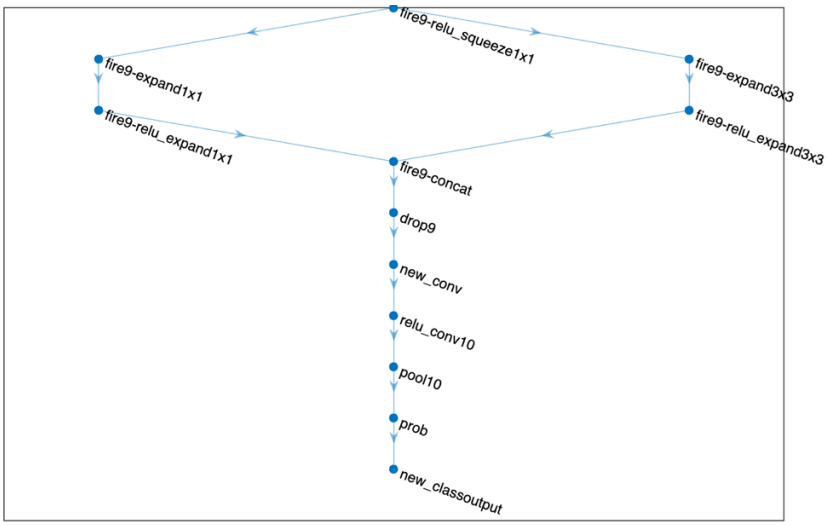
```
numClasses = numel(categories(imdsTrain.Labels));

if isa(learnableLayer,'nnet.cnn.layer.FullyConnectedLayer')
    newLearnableLayer = fullyConnectedLayer(numClasses, ...
        'Name','new_fc', ...
        'WeightLearnRateFactor',10, ...
        'BiasLearnRateFactor',10);

elseif isa(learnableLayer,'nnet.cnn.layer.Convolution2DLayer')
    newLearnableLayer = convolution2dLayer(1,numClasses, ...
        'Name','new_conv', ...
        'WeightLearnRateFactor',10, ...
        'BiasLearnRateFactor',10);
end

lgraph = replaceLayer(lgraph,learnableLayer.Name,newLearnableLayer);
```

Уровень классификации определяет выходные классы сети. Заменим классификационный слой на новый без меток классов. trainNetwork автоматически устанавливает выходные классы слоя во время обучения.

```
newClassLayer = classificationLayer('Name','new_classoutput');
lgraph = replaceLayer(lgraph,classLayer.Name,newClassLayer);
```

Чтобы убедиться, что новые слои соединены правильно, построим график нового слоя и увеличим последние слои сети.

```
figure('Units','normalized','Position',[0.3 0.3 0.4 0.4]);
plot(lgraph)
ylim([0,10])
```



## 4) Заморозка последних слоев

Теперь сеть готова к переобучению на новом наборе изображений. При желании мы можем «заморозить» веса более ранних слоев в сети, установив скорости обучения в этих слоях равными нулю. Во время обучения trainNetwork не обновляет параметры замороженных слоев. Поскольку градиенты замороженных слоев не нужно вычислять, замораживание весов многих начальных слоев может значительно ускорить обучение сети. Если новый набор данных невелик, то замораживание более ранних слоев сети также может предотвратить переоснащение этих слоев новым набором данных.

Извлечем слои и соединения графа слоев и выберем, какие слои заморозить. В GoogLeNet первые 10 слоев образуют начальный «стебель» сети. Используйте вспомогательную функцию freezeWeights , чтобы установить скорость обучения на ноль в первых 10 слоях. Используйте вспомогательную функцию createLgraphUsingConnections , чтобы повторно соединить все слои в исходном порядке. График нового слоя содержит те же слои, но с нулевыми скоростями обучения предыдущих слоев.

```
layers = lgraph.Layers;
connections = lgraph.Connections;

layers(1:10) = freezeWeights(layers(1:10));
lgraph = createLgraphUsingConnections(layers,connections);
```

## 5) Тренировка сети

Сети требуются входные изображения размером 227 x 227 x 3, но изображения в хранилище данных изображений имеют разные размеры. Используем расширенное хранилище данных изображений для автоматического изменения размера обучающих изображений. Укажем дополнительные операции увеличения для выполнения на обучающих изображениях: случайным образом отразим обучающие изображения по

вертикальной оси и произвольно переместим их до 30 пикселей и масштабируйте их до 10% по горизонтали и вертикали. Увеличение данных поможет предотвратить переоснащение сети и запоминание точных деталей обучающих изображений.

```
pixelRange = [-30 30];
scaleRange = [0.9 1.1];
imageAugmenter = imageDataAugmenter( ...
    'RandXReflection',true, ...
    'RandXTranslation',pixelRange, ...
    'RandYTranslation',pixelRange, ...
    'RandXScale',scaleRange, ...
    'RandYScale',scaleRange);
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain, ...
    'DataAugmentation',imageAugmenter);
```

Чтобы автоматически изменить размер проверочных изображений без выполнения дальнейшего увеличения данных, используйте хранилище данных дополненного изображения без указания каких-либо дополнительных операций предварительной обработки.

```
augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValidation);
```

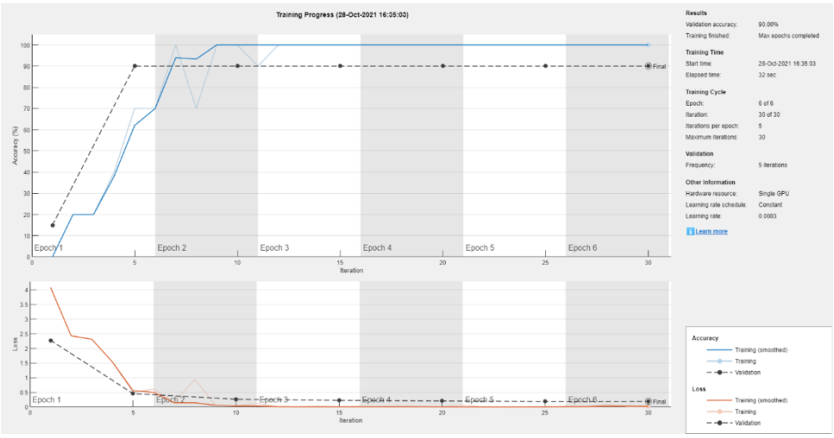
Укажем варианты обучения. Установим InitialLearnRate небольшое значение, чтобы замедлить обучение в переданных слоях, которые еще не заморожены. На предыдущем шаге вы увеличили коэффициенты скорости обучения для последнего обучаемого слоя, чтобы ускорить обучение на новых последних слоях. Эта комбинация настроек скорости обучения приводит к быстрому обучению на новых слоях, более медленному обучению на средних слоях и отсутствию обучения на более ранних, замороженных слоях.

Укажем количество эпох для обучения. Эпоха — это полный цикл обучения на всем наборе обучающих данных. Укажем размер мини-пакета и данные проверки. Вычислим точность проверки один раз за эпоху.

```
miniBatchSize = 10;
valFrequency = floor(numel(augimdsTrain.Files)/miniBatchSize);
options = trainingOptions('sgdm', ...
    'MiniBatchSize',miniBatchSize, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',3e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',valFrequency, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

Обучите сеть с помощью обучающих данных. По умолчанию trainNetwork использует графический процессор, если он доступен. Для этого требуется Parallel Computing Toolbox™ и поддерживаемое устройство GPU. В противном случае trainNetwork использует ЦП. Вы также можете указать среду выполнения, используя 'ExecutionEnvironment' аргумент пары "имя-значение" trainingOptions. Поскольку набор данных очень мал, обучение проходит быстро.

```
net = trainNetwork(augimdsTrain,lgraph,options);
```




## 6) Классификация проверочных изображения

Классифицируем проверочные изображения, используя точно настроенную сеть, и рассчитаем точность классификации.

```
[YPred,probs] = classify(net,augimdsValidation);
accuracy = mean(YPred == imdsValidation.Labels)
```

```
accuracy = 0.9000
```

Отобразим четыре образца проверочных изображений с предсказанными метками и предсказанными вероятностями изображений, имеющих эти метки.

```
idx = randperm(numel(imdsValidation.Files),4);
figure
for i = 1:4
    subplot(2,2,i)
    I = readimage(imdsValidation,idx(i));
    imshow(I)
    label = YPred(idx(i));
    title(string(label) + ", " + num2str(100*max(probs(idx(i),:)),3) + "%");
end
```

*Результат:*