DL: Передача обучения

Передача обучения

Пред обученная CNN (squeezeNet – 1000 классов)

Яблочный пирог

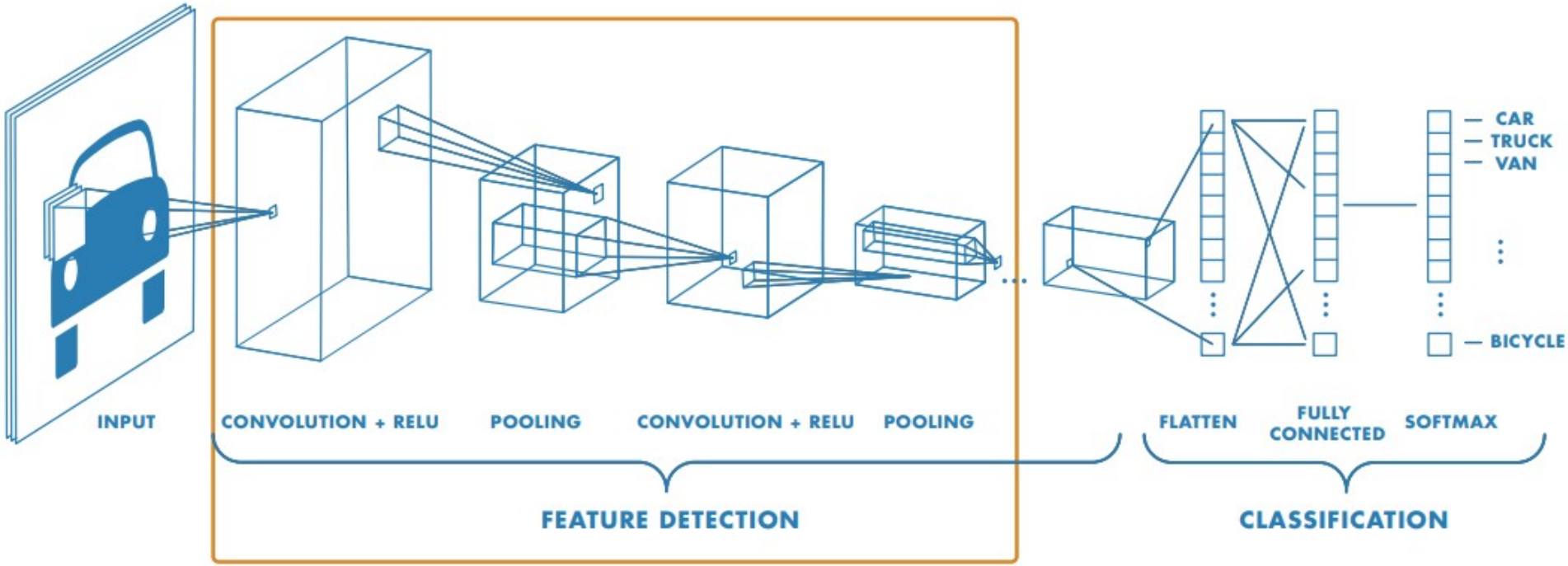
Карпаччо

Сырная тарелка

Яичница

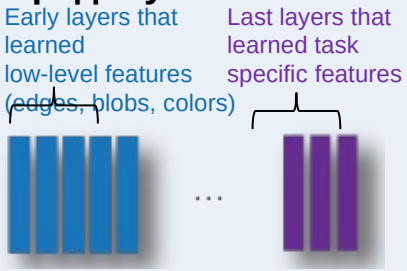
Новая задача – 4 класса для классификации

Передача обучения



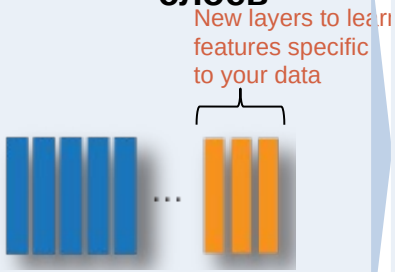
Передача обучения

Выбор и загрузка предобученной сети



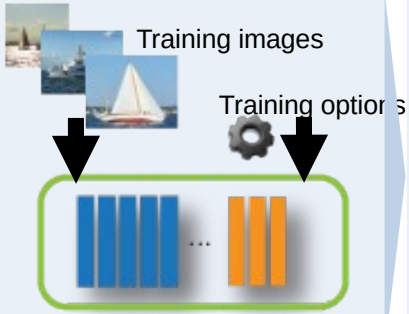
1 миллион изображений
1000 классов

Замена последних слоев



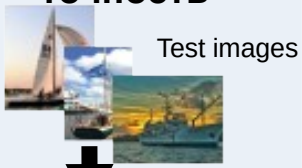
Несколько классов
Обучаются быстрее

Обучение сети

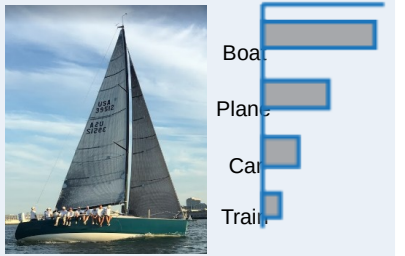


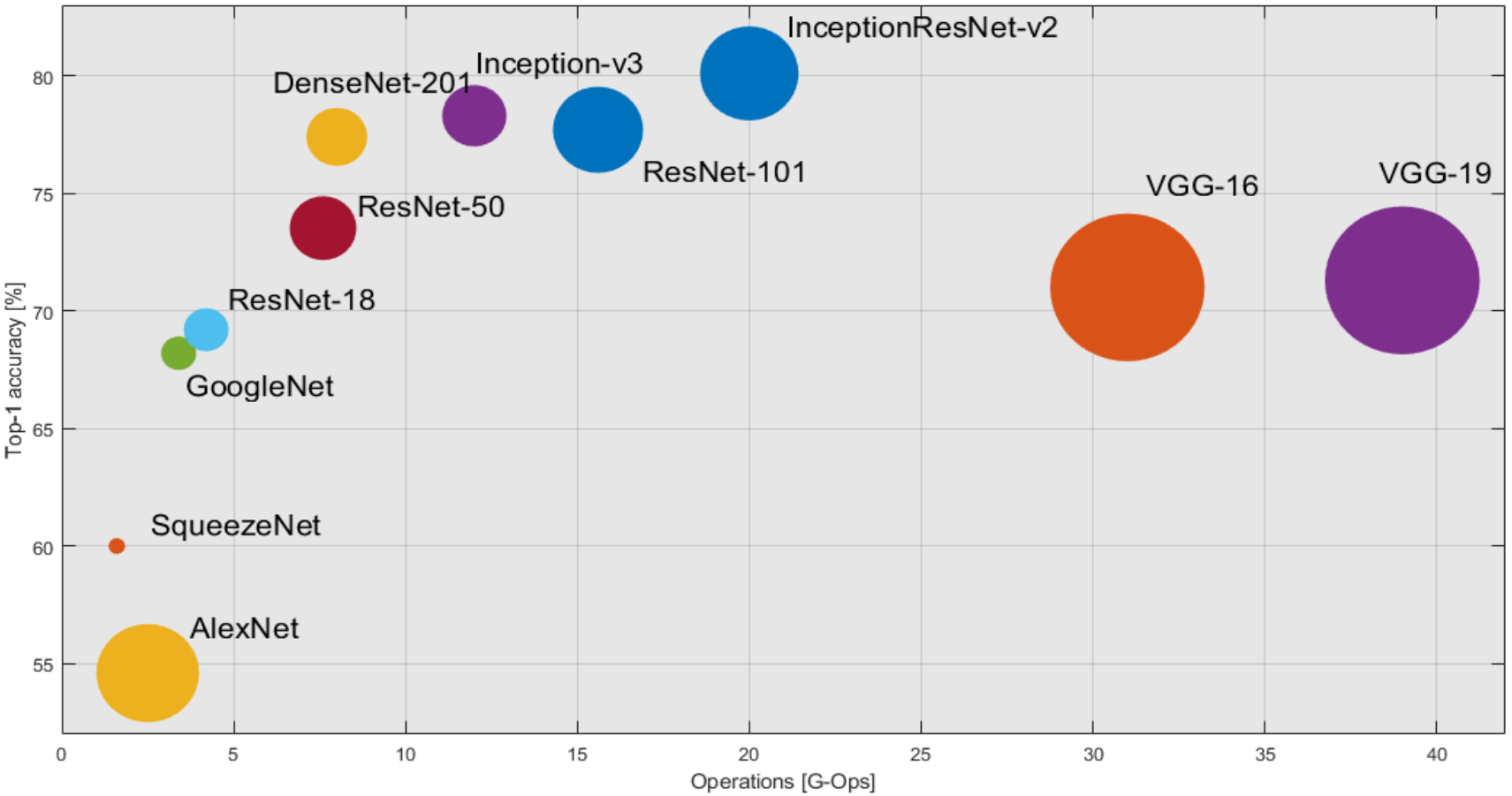
100и изображений
10и классов

Предсказание и точность

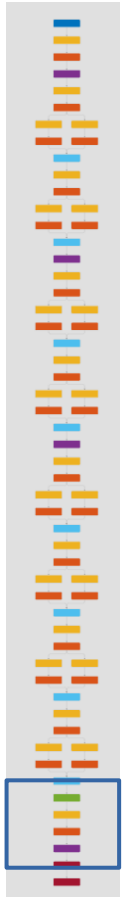


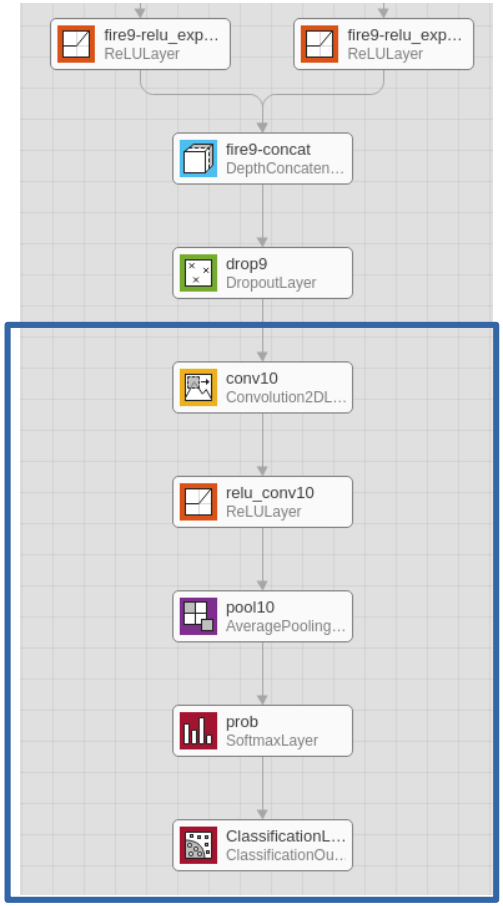
Развертывание результатов

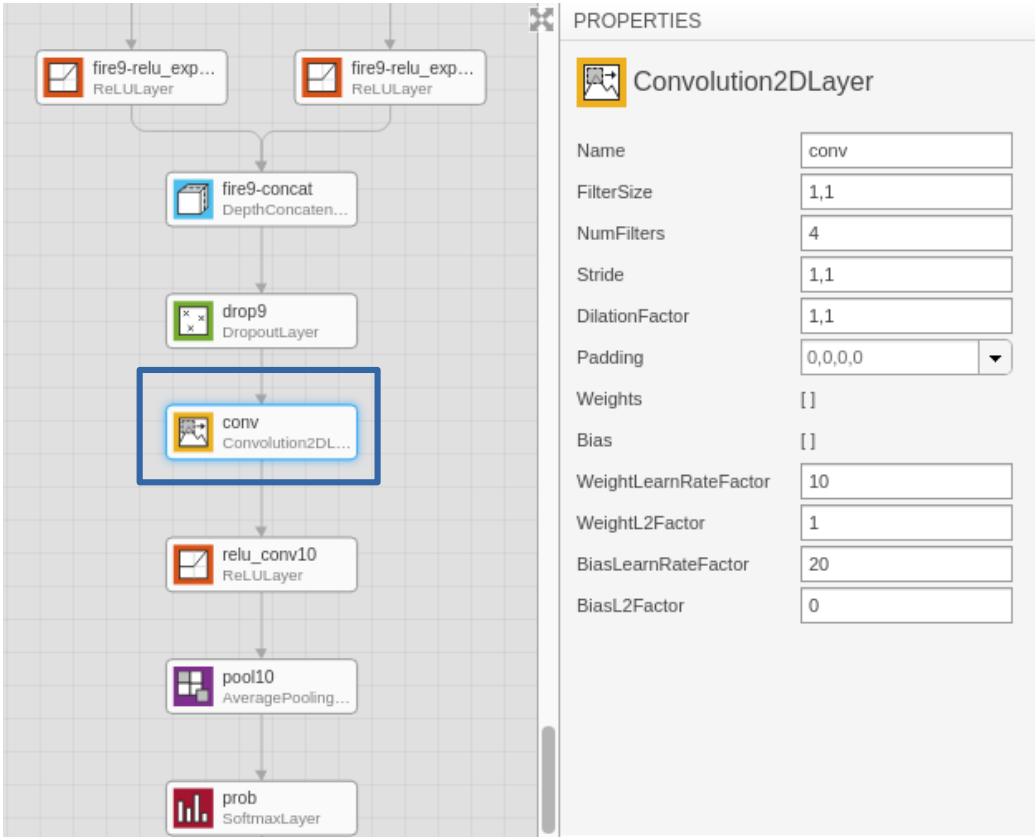



InceptionResNet-v2
Inception-v3
DenseNet-201
ResNet-101
VGG-16
VGG-19
ResNet-50
ResNet-18
GoogleNet
SqueezeNet
AlexNet
Top-1 accuracy [%]
Operations [G-Ops]
80
75
70
65
60
55
0
5
10
15
20
25
30
35
40

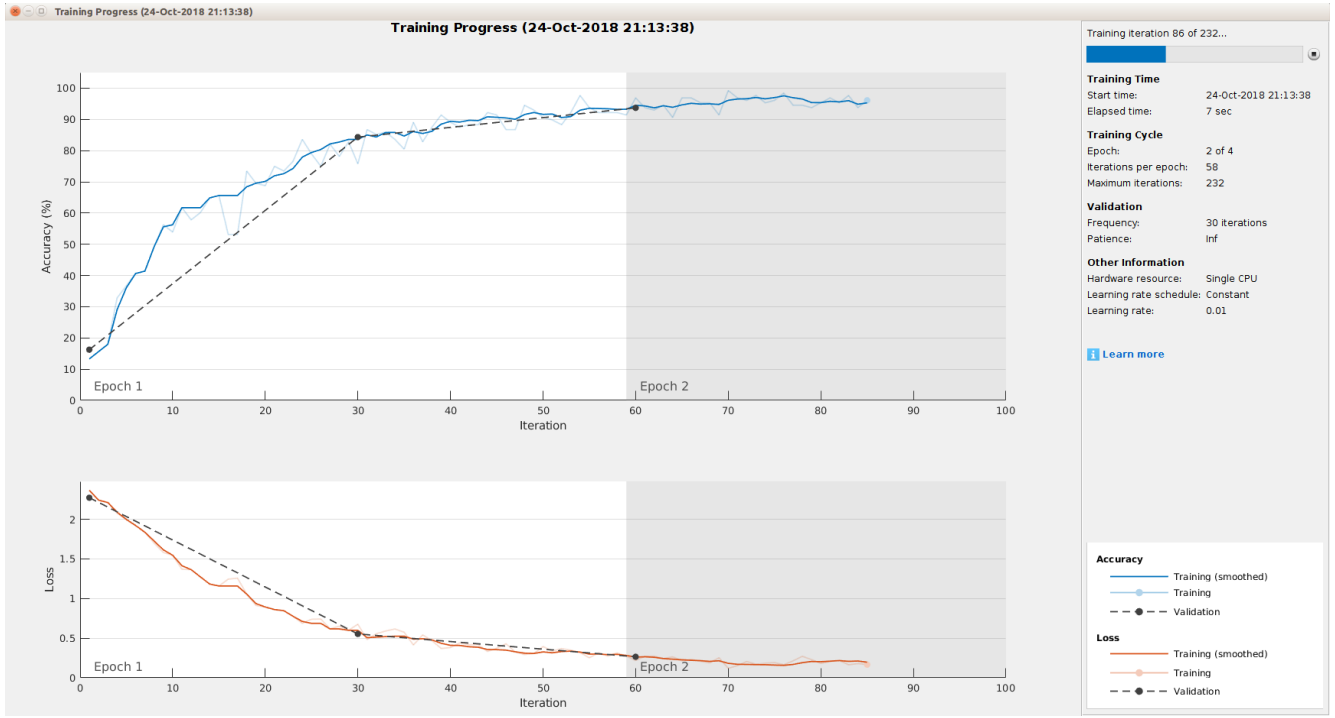
Замена последних слоев







Обучение: все слои или последние



Результаты

	• SqueezeNet	• SqueezeNet (2 слоя)	• VGG16	• InceptionResNet	Alexnet
Время обучения (мин)	7:23	7:34	14:16	112:16	12:44
Точность (%)	92	93.5	94	91.5	88.7
Время классификации (сек)	4.0	4.0	5.4	14.82	4.4
Вес (МБ)	6.6	6.6	538	226	245

Пример: VGG19

```
VGG(
  (features): Sequential(
          (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (1): ReLU(inplace)
          (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (3): ReLU(inplace)
---------------------------------------------------------------------------------
          (32): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (33): ReLU(inplace)
          (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (35): ReLU(inplace)
          (36): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1)

)
  (classifier): Sequential(
          (0): Linear(in_features=25088, out_features=4096, bias=True)
          (1): ReLU(inplace)
          (2): Dropout(p=0.5)
          (3): Linear(in_features=4096, out_features=4096, bias=True)
          (4): ReLU(inplace)
          (5): Dropout(p=0.5)
          (6): Linear(in_features=4096, out_features=2, bias=True)
```

Пример VGG19

```
vgg_based = torchvision.models.vgg19(pretrained=True)
## freeze the layers
for param in vgg_based.parameters():
   param.requires_grad = False

# Modify the last layer
number_features = vgg_based.classifier[6].in_features
features = list(vgg_based.classifier.children())[:-1] # Remove last layer
features.extend([torch.nn.Linear(number_features, len(class_names))])
vgg_based.classifier = torch.nn.Sequential(*features)

vgg_based = vgg_based.to(device)

print(vgg_based)

criterion = torch.nn.CrossEntropyLoss()
optimizer_ft = optim.SGD(vgg_based.parameters(), lr=0.001, momentum=0.9)
```