БДЗ1 DL Аксёнов Ярослав

18 января 2023 г.

1 Самодельное подобие резнета

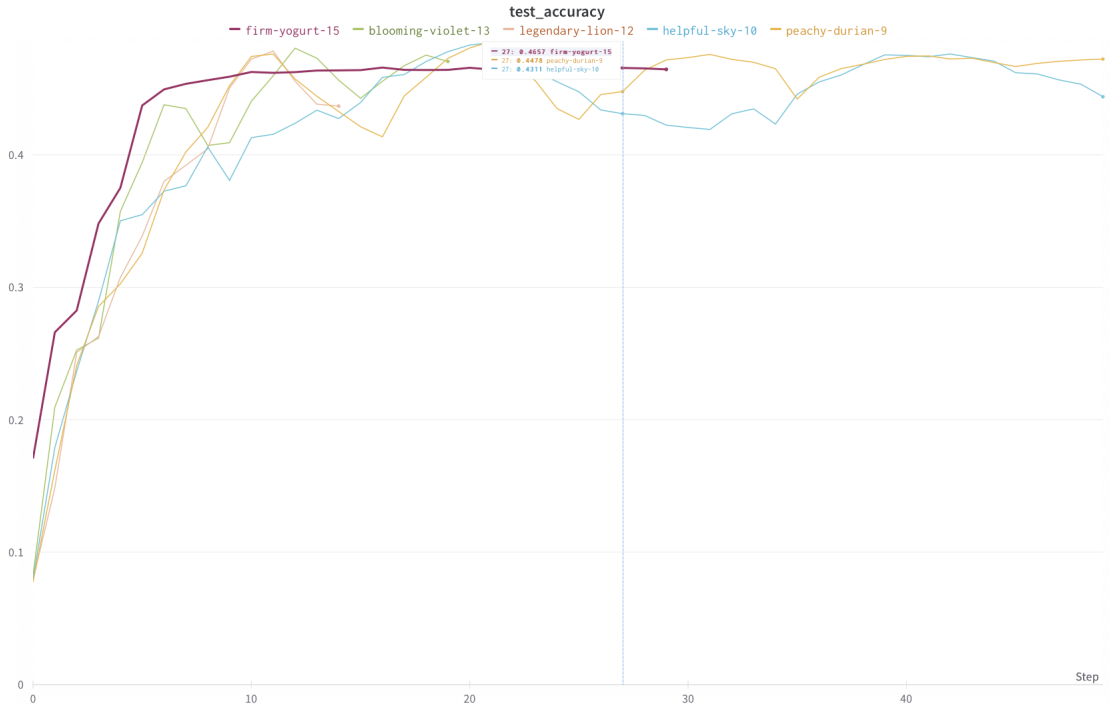
Для начала я использовал 6 блоков резнета из домашнего задания по глубинному обучению. Структура была следующая (лучший результат оказался при 6 блоках, так что картинка с ним)

```python
class MiniResNet(nn.Module):
    def __init__(self):
        super().__init__()

        self.activation = nn.Sequential(
            nn.ReLU(),
            nn.MaxPool2d(2)
        )

        self.conv1 = MiniResNet.conv_block(3, 32)
        self.skip1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=1)

        self.conv2 = MiniResNet.conv_block(32, 64)
        self.skip2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=1)

        self.conv3 = MiniResNet.conv_block(64, 128)
        self.skip3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=1)

        self.conv4 = MiniResNet.conv_block(128, 256)
        self.skip4 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=1)

        self.conv5 = MiniResNet.conv_block(256, 512)
        self.skip5 = nn.Conv2d(in_channels=256, out_channels=512, kernel_size=1)

        self.conv6 = MiniResNet.conv_block(512, 1024)
        self.skip6 = nn.Conv2d(in_channels=512, out_channels=1024, kernel_size=1)

        self.head = nn.Sequential(
            nn.Linear(in_features=9216, out_features=4096),
            nn.BatchNorm1d(4096),
            nn.ReLU(),
            nn.Linear(in_features=4096, out_features=CLASS_QUANTITY)
        )

```

```python
@staticmethod
def conv_block(in_channels, out_channels):
    return nn.Sequential(
        nn.Conv2d(in_channels=in_channels, out_channels=out_channels, kernel_size=3, padding='same'),
        nn.BatchNorm2d(out_channels),
        nn.ReLU(),
        nn.Conv2d(in_channels=out_channels, out_channels=out_channels, kernel_size=3, padding='same'),
        nn.BatchNorm2d(out_channels)
    )
```

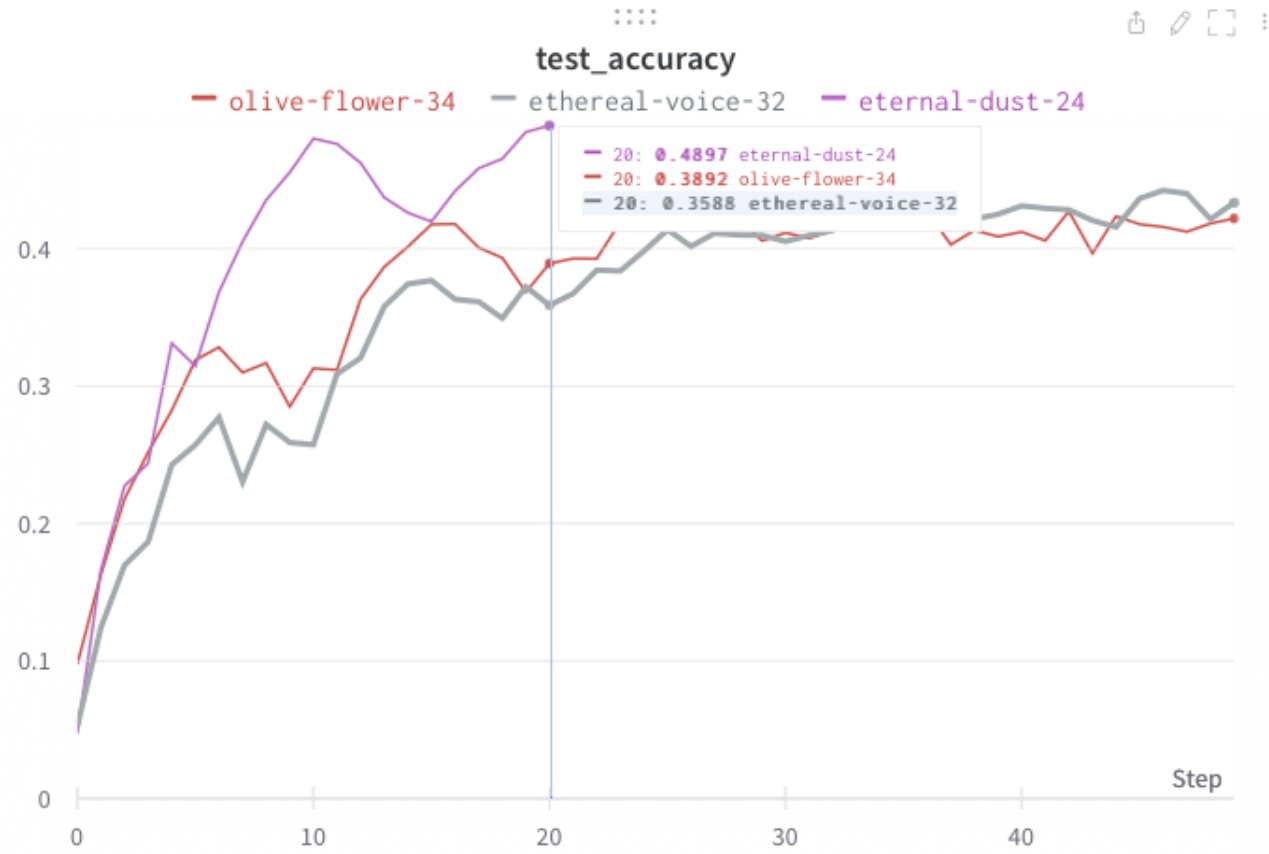
```python
def forward(self, x):
    x = self.activation(self.conv1(x) + self.skip1(x))
    x = self.activation(self.conv2(x) + self.skip2(x))
    x = self.activation(self.conv3(x) + self.skip3(x))
    x = self.activation(self.conv4(x) + self.skip4(x))
    x = self.activation(self.conv5(x) + self.skip5(x))
    x = self.activation(self.conv6(x) + self.skip6(x))
    x = torch.flatten(x, 1)
    x = self.head(x)
    return x
```

Везде используется нормализация по данным ImageNet, rotate(-10, 10) и random horizontal flip с вероятностью 0.5. Провел эксперименты с разными расписаниями learning rate для 6-ти блочного резнета. Один с CyclicLR(base_lr=1e-4, max_lr=0.1, step_size_up=10, mode=triangular2), другой с step_size_up=5, используя при этом SGD и SGD+momentum. С моментом очевидно дал более быструю сходимость и выше результатат, поэтому оставляю только графики с ним. blooming-violet с CyclicLR(base-lr=1e-4, max-lr=1e-2, step-size-up=3, mode=triangular2). На графике legendary-lion с CyclicLR(base-lr=1e-4, max-lr=1e-1, step-size-up=5, mode=triangular2), firm-yogurt с MultiStepLR(milestones=[5, 10, 20], gamma=0.1). Multistep запускал на 30 эпох и получи точность меньше чем у лучшего с циклическим lr за 20 эпох:

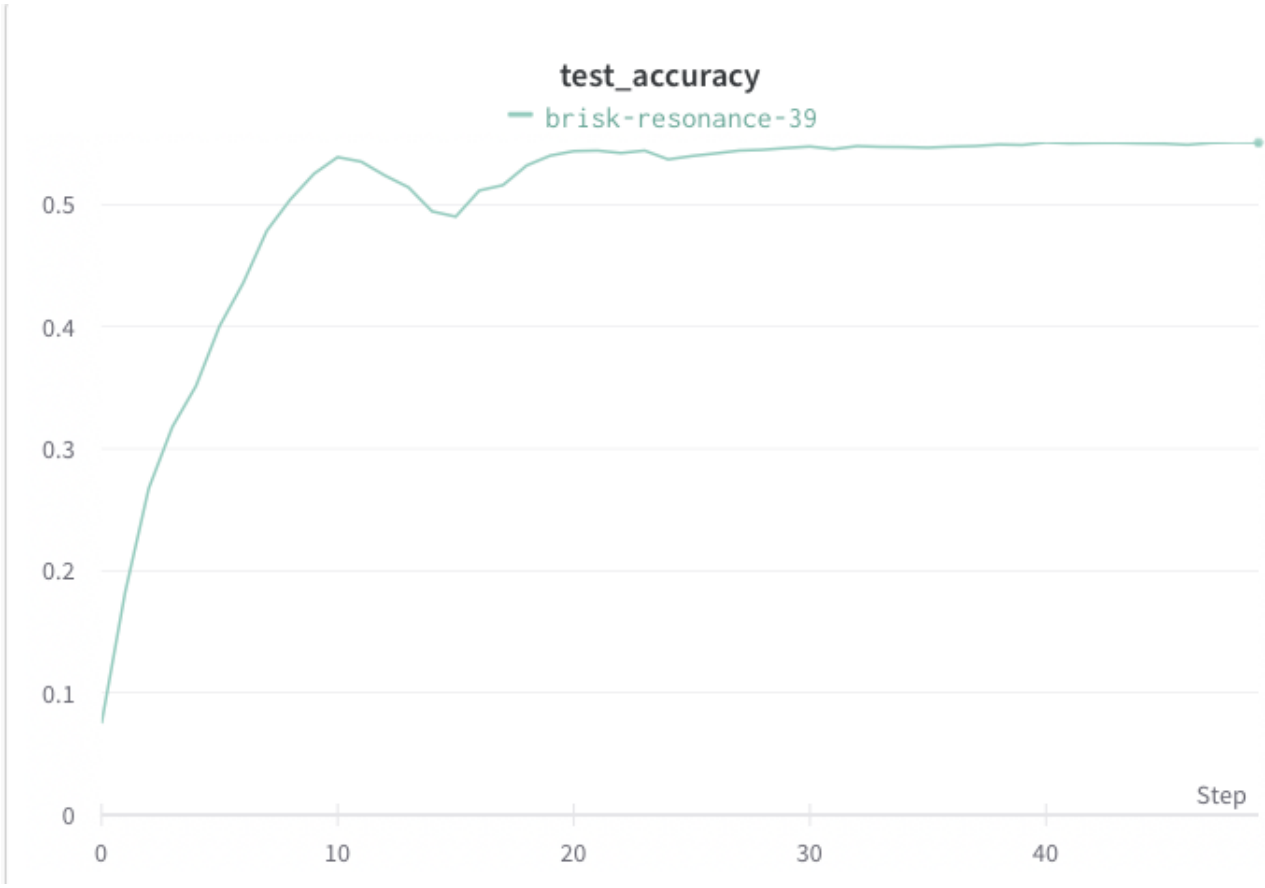


Лучший результат 48.97 процента точности (который был отправлен в качестве первого чекпоинта был получен за 21 эпоху обучения с помощью CyclicLR(base-lr=1e-4, max-lr=0.1, step-size-up=5, mode=triangular2)), это eternal-dust.

Далее решил добавить в полносвзяный блок дропаут, то есть голова выглядела как dropout, linear, batchnorm, reLU, dropout, linear. Провел два эксперимента с вероятностями 0.5 (ethereal-voice) и 0.2 (olive-flower), получил результаты хуже, поэтому дропаут больше не использовал.

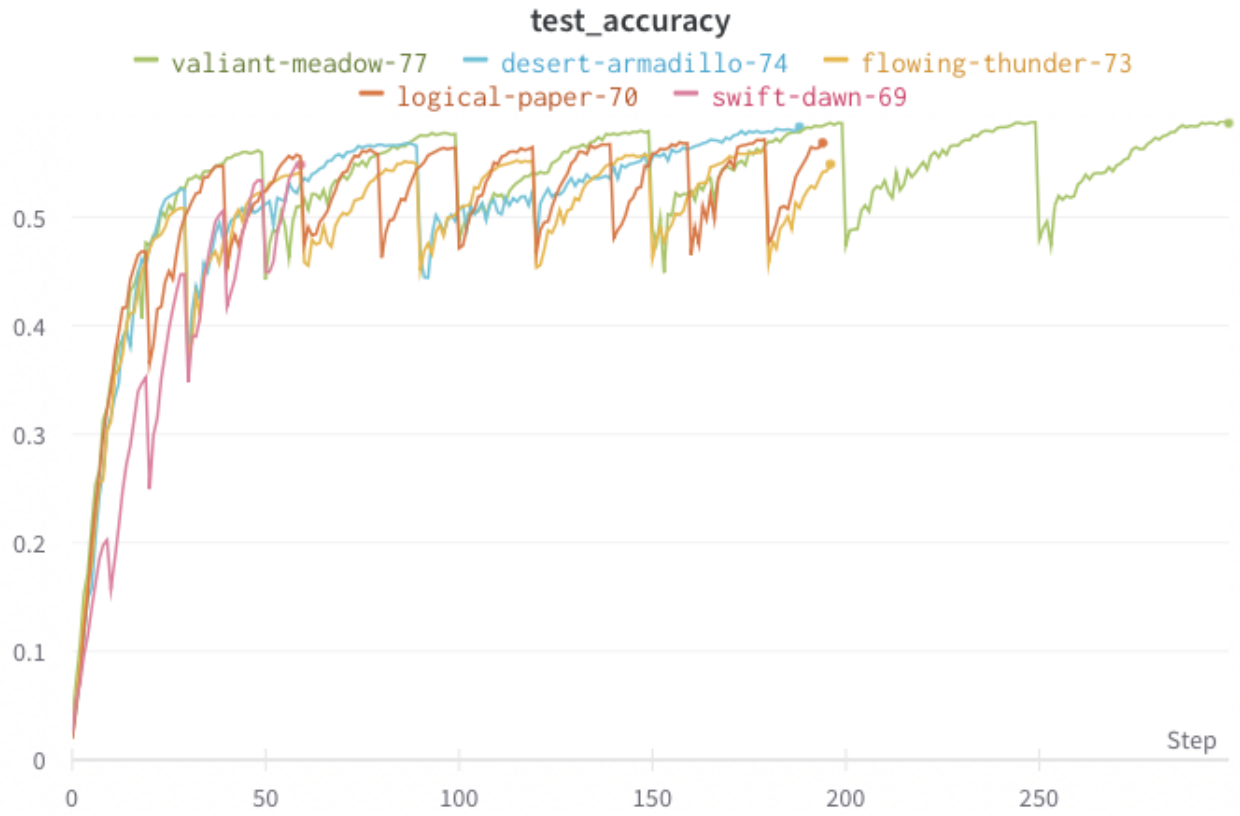


Далее я сообразил, что ресайз увеличит размерности сверток, сооответственно увеличит количество признаков и потенциально может улучшить качество модели, поэтому сделалл ресайз картинки с 64 на 64 до 224 на 224 (стандратный размер резнета). Это улучшило точность классификации до 55 процентов:



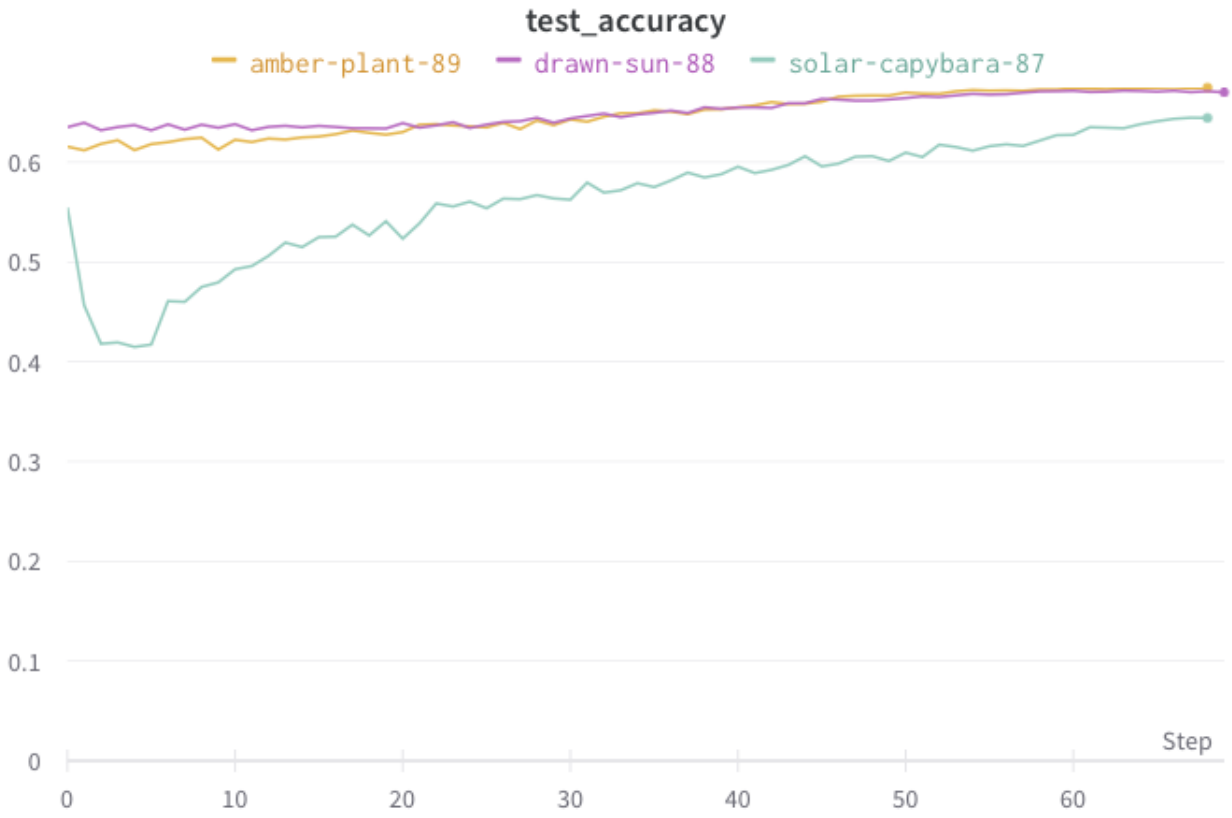
Я перебрал разное количество блоков (от 6, 7 и 9) и получил, что увеличение до 7 и 9 не дало особого резульлата (я обучал всего 50 эпох, поэтому возможно модель с большим числом параметров и могла бы перегнать болоее простую, но для этого необходимо было бы больше эпох).

Дальше я решил, что пора переходить к более тяжелой артиллерии и взял ResNext50 32x4d, а также добавил аугментацию RandomResizedCrop со скейлом (0.5, 1) и (0.2, 1) и воспользовался схемой предложенной в статье про быстрое обучение резнета, а именноо расписанием CosineAnnealingWarmRestarts с вормапом каждые 30 эпох и поставил обучаться 300 эпох, результаты стали всего немного выше (надо было оставить базовый скейл 0.08, с ним получился лучший результат в последующих экспериментах с другим lr sheduler). Получились следующие графики, разные частоты вормапов и количество эпох, но в целоом резульаты улучшить особо не удалось (лишь до 59 на valiant-meadow).



Далее прооизошло отключение датасферы и у меня остался только кагл на 30 часов в неделю с максимальной длиной одного запуска 12 часов, поэтому я несколько раз дообучивал получившуюся тут модель сначала с помощью CosineAnnealing (с 0.5 до нуля) с линейным вормапом первые 5 эпох, затем просто CosineAnnealing в зависимости от того lr на котором остановился (так как не мог поставить сразу 300 эпох условно обучаться обучал по 68 эпох, сколько максимально влезало в 12 часов). Кроме того я добавил побольше аугментаций помимо имеющихся, а именно TrivialAugmentWide(), который сочетает в себе разные аугментации цвета, поворота и искажения перспективы, и RandomErasing(p=0.1), который с вероятностью 10 процентов вырезает из картинки некоторый участок (помогает от переобучения). Такой сетап помог после 3 последовательных запусков дообучения достичь качества на валидации в 0.6746, что на тесте дало 0.6532. Также добавил weight-decay на веса, кроме bias и батч нормы, label-smoothing=0.1 и стал делать ресайз на инференсе

232 (прочитал в инете, что так лучше рецептивное поле работает).



Голубой график соответственно первый запуск. Видим, что из за вормапа аккураси просел (я подумал, что из-за ввода новых моощных аугментаций надо дать модели выйти из локального минимума), но затеме вырос до 64 процентов. Последующие два запуска позволили увеличить акураси до 67 и затем до 67.4 процентов.