Лабораторная работа №2

по дисциплине: Технологии нейросетевых вычислений

вариант: Plant Segmentation

Выполнили: Миху Вадим

Чернова Анна

P34301

Преподаватель: Старобыховская Анастасия Александровна

Санкт-Петербург
2024

Задание

1. Выполнить Лабораторную работу 1
2. Согласовать способ оптимизации для вашей модели (см слайд 6)
3. Реализовать способ
a. Можно использовать функции и модули pyTorch
4. Получить метрики и графики обучения, инференса
5. Выполнить анализ и сравнение с исходной моделью
6. Получить оценки по точности, примеры ошибочных семплов, сделать выводы
7. Создать отчет (добавить графики лосса, точности и др.), прикрепить к заданию
8. Защититься на 12+ баллов

Выполнение

I. Модель

Исходный код модели

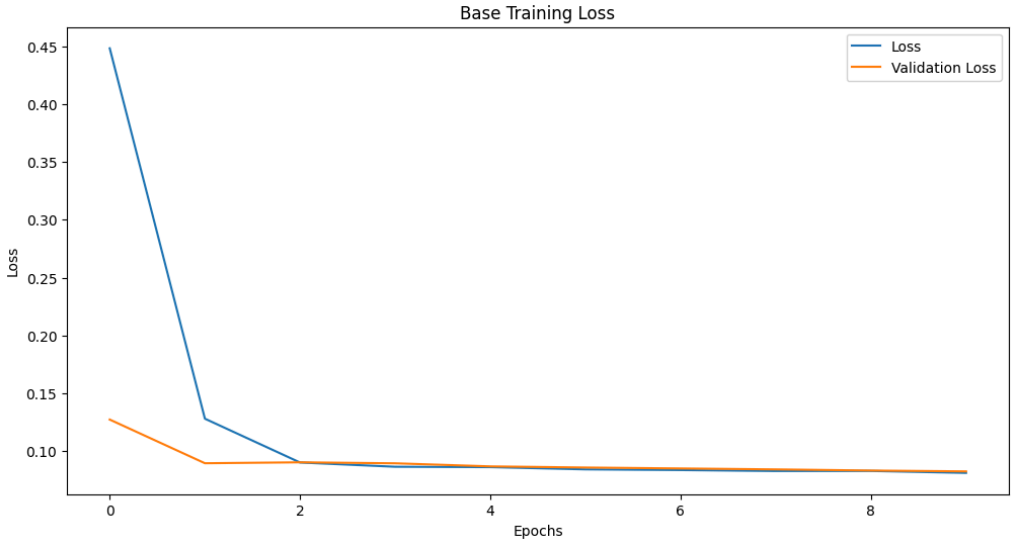
```
def create_unet_model(input_size=(IMG_HEIGHT, IMG_WIDTH, 3)):
    inputs = keras.layers.Input(input_size)

    c1 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(inputs)
    c1 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(c1)
    p1 = keras.layers.MaxPooling2D((2, 2))(c1)

    c2 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(p1)
    c2 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(c2)
    p2 = keras.layers.MaxPooling2D((2, 2))(c2)

    c3 = keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same')(p2)
    c3 = keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same')(c3)

    u1 = keras.layers.UpSampling2D((2, 2))(c3)
    u1 = keras.layers.concatenate([u1, c2])
    c4 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(u1)
    c4 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(c4)

    u2 = keras.layers.UpSampling2D((2, 2))(c4)
    u2 = keras.layers.concatenate([u2, c1])
    c5 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(u2)
    c5 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(c5)

    outputs = keras.layers.Conv2D(1, (1, 1), activation='sigmoid')(c5)
    model = keras.Model(inputs, outputs)
    return model
```

II. Оптимизация

Метрики не обученной модели без оптимизации.

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	[(None, 256, 256, 3)]	0	[]
conv2d_44 (Conv2D)	(None, 256, 256, 64)	1792	['input_5[0][0]']
conv2d_45 (Conv2D)	(None, 256, 256, 64)	36928	['conv2d_44[0][0]']
max_pooling2d_8 (MaxPooling2D)	(None, 128, 128, 64)	0	['conv2d_45[0][0]']
conv2d_46 (Conv2D)	(None, 128, 128, 128)	73856	['max_pooling2d_8[0][0]']
conv2d_47 (Conv2D)	(None, 128, 128, 128)	147584	['conv2d_46[0][0]']
max_pooling2d_9 (MaxPooling2D)	(None, 64, 64, 128)	0	['conv2d_47[0][0]']
conv2d_48 (Conv2D)	(None, 64, 64, 256)	295168	['max_pooling2d_9[0][0]']
conv2d_49 (Conv2D)	(None, 64, 64, 256)	590080	['conv2d_48[0][0]']
up_sampling2d_8 (UpSampling2D)	(None, 128, 128, 256)	0	['conv2d_49[0][0]']
concatenate_8 (Concatenate)	(None, 128, 128, 384)	0	['up_sampling2d_8[0][0]', 'conv2d_47[0][0]']
conv2d_50 (Conv2D)	(None, 128, 128, 128)	442496	['concatenate_8[0][0]']
conv2d_51 (Conv2D)	(None, 128, 128, 128)	147584	['conv2d_50[0][0]']
up_sampling2d_9 (UpSampling2D)	(None, 256, 256, 128)	0	['conv2d_51[0][0]']
concatenate_9 (Concatenate)	(None, 256, 256, 192)	0	['up_sampling2d_9[0][0]', 'conv2d_45[0][0]']
conv2d_52 (Conv2D)	(None, 256, 256, 64)	110656	['concatenate_9[0][0]']
conv2d_53 (Conv2D)	(None, 256, 256, 64)	36928	['conv2d_52[0][0]']
conv2d_54 (Conv2D)	(None, 256, 256, 1)	65	['conv2d_53[0][0]']

Total params: 1883137 (7.18 MB)
Trainable params: 1883137 (7.18 MB)
Non-trainable params: 0 (0.00 Byte)

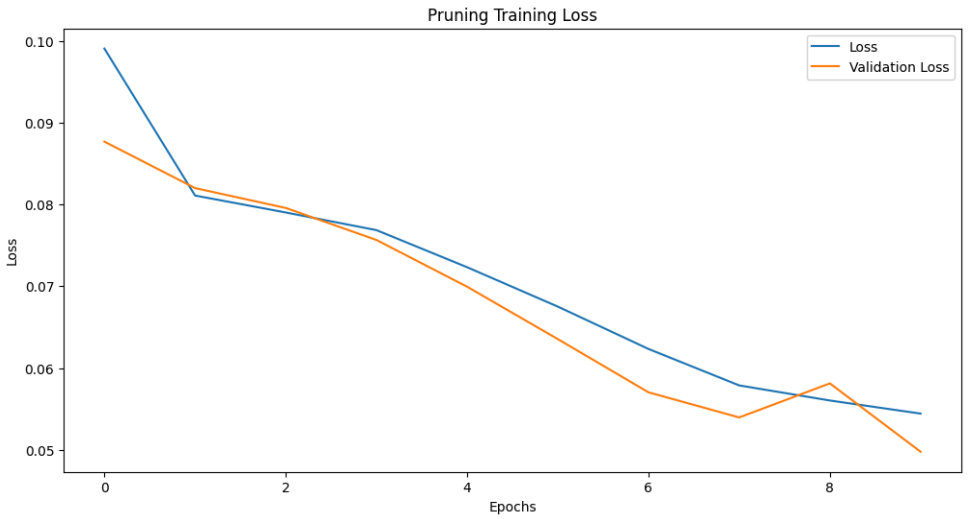


Применим к модели prune_low_magnitude прунинг, после чего обучим ее до конца.

```
prune_low_magnitude = tfmot.sparsity.keras.prune_low_magnitude
pruning_schedule = tfmot.sparsity.keras.PolynomialDecay(
    initial_sparsity=0.2,
    final_sparsity=0.8,
    begin_step=0,
    end_step=1000
)

pruned_model = prune_low_magnitude(base_model, pruning_schedule)

pruned_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

callbacks = [
    tfmot.sparsity.keras.UpdatePruningStep(),
    tfmot.sparsity.keras.PruningSummaries(log_dir='./logs')
]
```

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_5 (InputLayer)         [(None, 256, 256, 3)]     0         []

prune_low_magnitude_conv2d  (None, 256, 256, 64)      3522      ['input_5[0][0]']
_44 (PruneLowMagnitude)

prune_low_magnitude_conv2d  (None, 256, 256, 64)      73794     ['prune_low_magnitude_conv2d_4
_45 (PruneLowMagnitude)                                         4[0][0]']

prune_low_magnitude_max_po  (None, 128, 128, 64)      1         ['prune_low_magnitude_conv2d_4
oling2d_8 (PruneLowMagnitu                                      5[0][0]']
de)

prune_low_magnitude_conv2d  (None, 128, 128, 128)     147586    ['prune_low_magnitude_max_pool
_46 (PruneLowMagnitude)                                         ing2d_8[0][0]']

prune_low_magnitude_conv2d  (None, 128, 128, 128)     295042    ['prune_low_magnitude_conv2d_4
_47 (PruneLowMagnitude)                                         6[0][0]']

prune_low_magnitude_max_po  (None, 64, 64, 128)       1         ['prune_low_magnitude_conv2d_4
oling2d_9 (PruneLowMagnitu                                      7[0][0]']
de)

prune_low_magnitude_conv2d  (None, 64, 64, 256)       590082    ['prune_low_magnitude_max_pool
_48 (PruneLowMagnitude)                                         ing2d_9[0][0]']

prune_low_magnitude_conv2d  (None, 64, 64, 256)       1179906   ['prune_low_magnitude_conv2d_4
_49 (PruneLowMagnitude)                                         8[0][0]']

prune_low_magnitude_up_sam  (None, 128, 128, 256)     1         ['prune_low_magnitude_conv2d_4
pling2d_8 (PruneLowMagnitu                                      9[0][0]']
de)

prune_low_magnitude_concat  (None, 128, 128, 384)     1         ['prune_low_magnitude_up_sampl
enate_8 (PruneLowMagnitude                                      ing2d_8[0][0]',
)                                                                'prune_low_magnitude_conv2d_4
                                                                7[0][0]']

prune_low_magnitude_conv2d  (None, 128, 128, 128)     884866    ['prune_low_magnitude_concaten
_50 (PruneLowMagnitude)                                         ate_8[0][0]']

prune_low_magnitude_conv2d  (None, 128, 128, 128)     295042    ['prune_low_magnitude_conv2d_5
_51 (PruneLowMagnitude)                                         0[0][0]']

prune_low_magnitude_up_sam  (None, 256, 256, 128)     1         ['prune_low_magnitude_conv2d_5
pling2d_9 (PruneLowMagnitu                                      1[0][0]']
de)

prune_low_magnitude_concat  (None, 256, 256, 192)     1         ['prune_low_magnitude_up_sampl
enate_9 (PruneLowMagnitude                                      ing2d_9[0][0]',
)                                                                'prune_low_magnitude_conv2d_4
                                                                5[0][0]']

prune_low_magnitude_conv2d  (None, 256, 256, 64)      221250    ['prune_low_magnitude_concaten
_52 (PruneLowMagnitude)                                         ate_9[0][0]']

prune_low_magnitude_conv2d  (None, 256, 256, 64)      73794     ['prune_low_magnitude_conv2d_5
_53 (PruneLowMagnitude)                                         2[0][0]']

prune_low_magnitude_conv2d  (None, 256, 256, 1)       131       ['prune_low_magnitude_conv2d_5
_54 (PruneLowMagnitude)                                         3[0][0]']

==================================================================================================
Total params: 3765021 (14.36 MB)
Trainable params: 1883137 (7.18 MB)
Non-trainable params: 1881884 (7.18 MB)
```

Прунинг применяется ко всем слоям модели. Ниже приведен график падения лосса при обучении.



III. Анализ

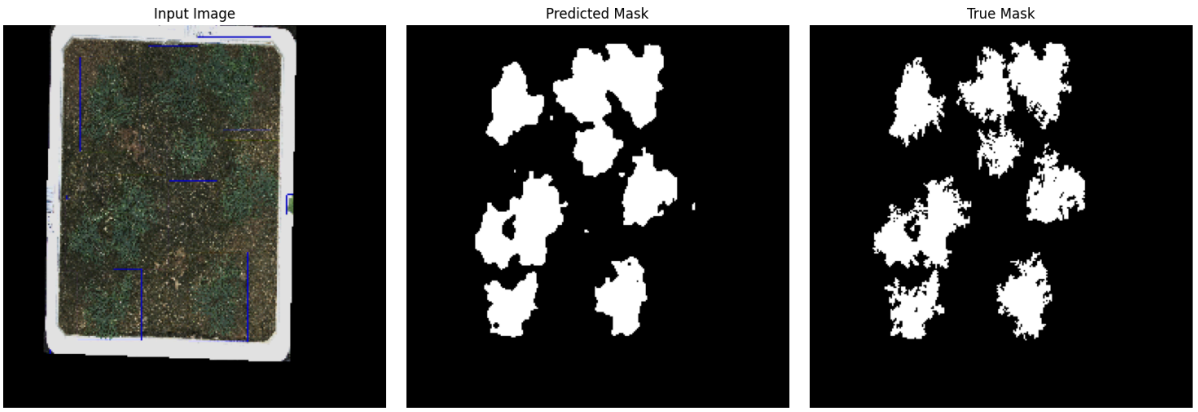
Из метрик и графиков обучения можно сделать предположение о том, что дальнейшее обучение происходит эффективно, так как метрики улучшаются.

Произведем подсчет непустых нейронов модели, чтобы понять, какую часть из них занулил прунинг.

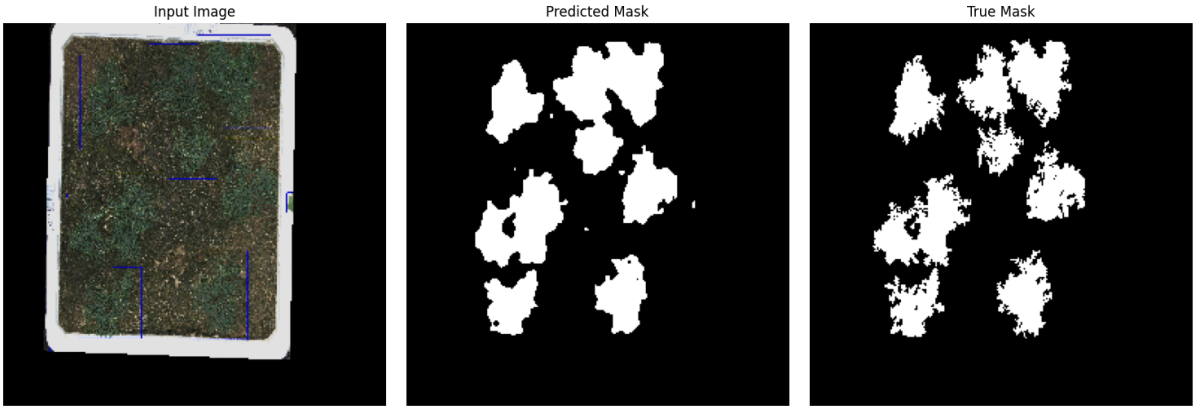
```
Number of non-null neurons in the base model: 1883137
Number of non-null neurons in the pruned model: 1385988
```

IV. Сравнение

Предсказание оригинальной модели. Выбрана случайная картинка из тренировочных данных.



Предсказание модели с прунингом. Как можем наблюдать, разница на глаз неотличима.



V. Вывод

Использование прунинга для оптимизации модели является хорошим решением. Небольшой процент прунинга фактически не имеет влияния