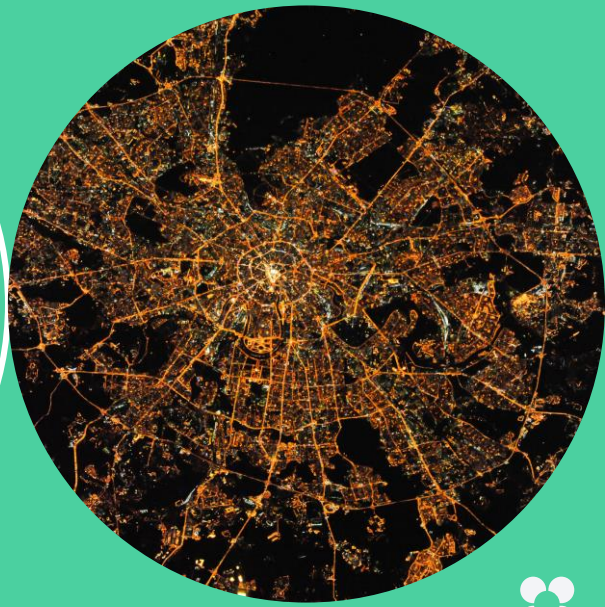
Семантическая сегментация спутниковых снимков

Анна Семина

ПОСТАНОВКА ЗАДАЧИ

1



Постановка задачи

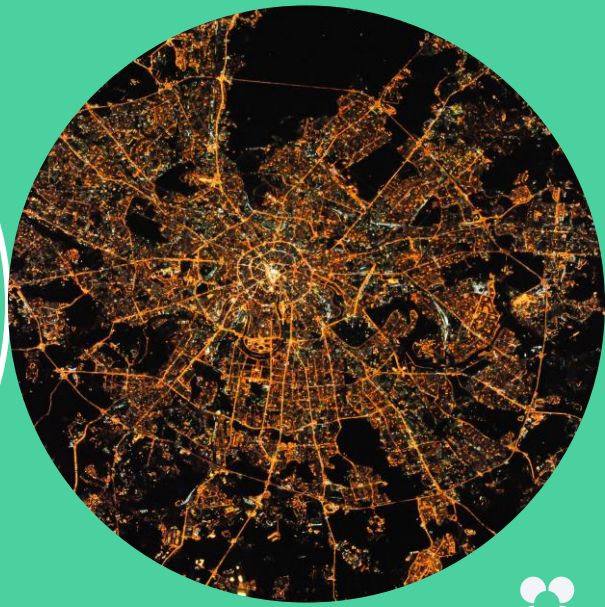
Цель проекта – построение модели на базе сверточных нейронных сетей для распознавания застроенных территорий на спутниковых снимках различных городов.

План решения задачи

1. Анализ литературы
2. Анализ исходных данных
3. Предобработка данных
4. Построение моделей
5. Сравнение результатов
6. Выводы

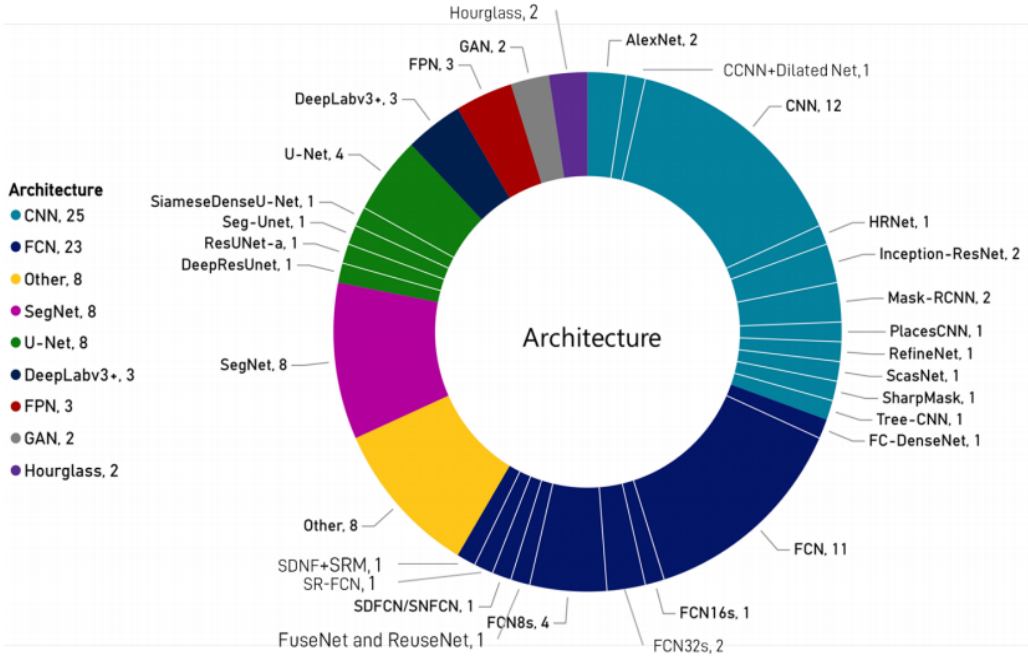
АНАЛИЗ ЛИТЕРАТУРЫ

2



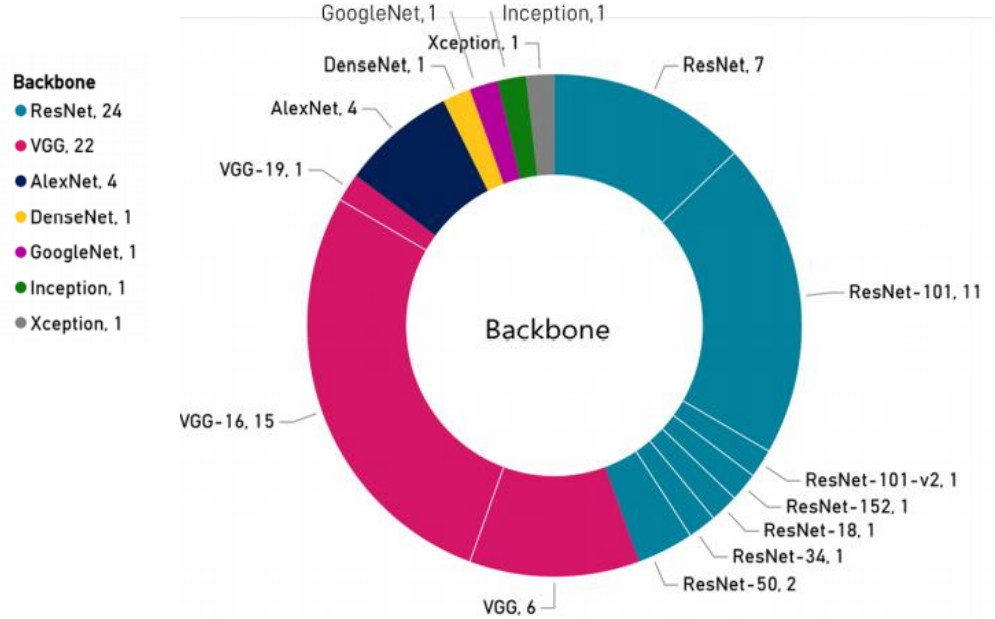
Обзор используемых DL архитектур

Наиболее часто в задачах сегментации спутниковых снимков использовались модели FCN, U-Net, SegNet.



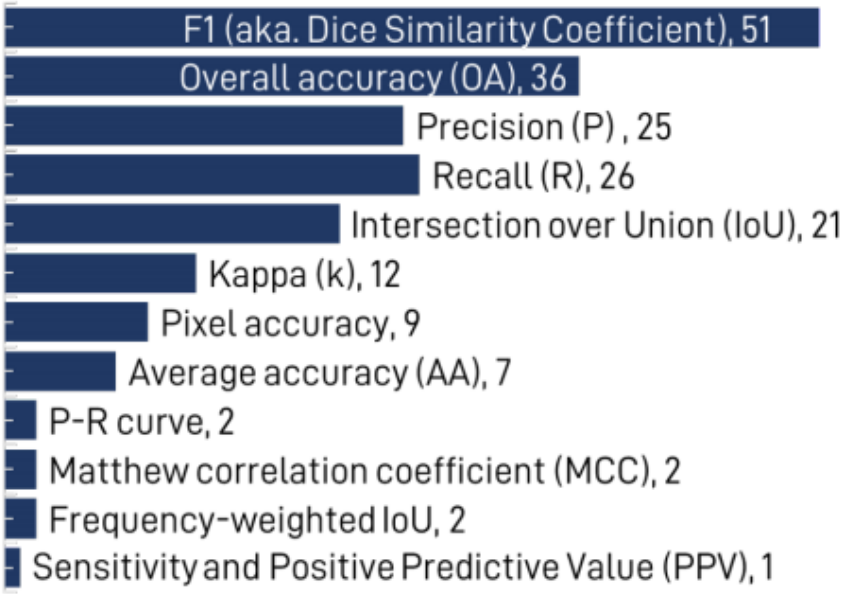
Обзор базовых CN

Наиболее часто используются ResNet и VGG.



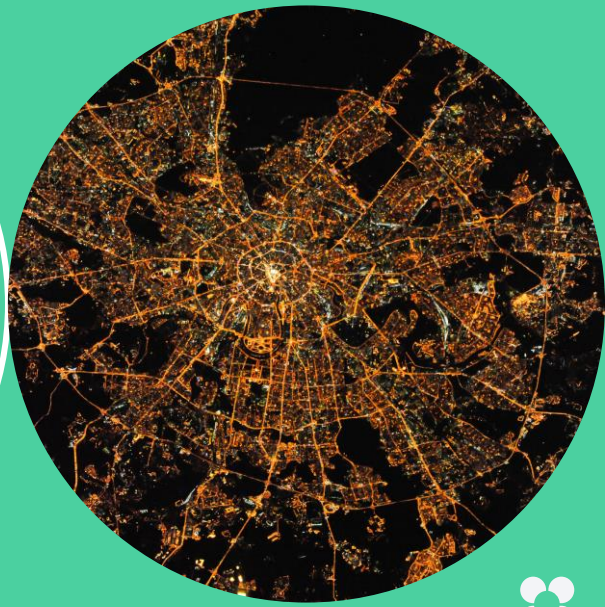
Обзор используемых метрик

Наиболее часто использовались коэффициент Дайса (F1/Dice) и точность (Accuracy).



АНАЛИЗ ДАННЫХ

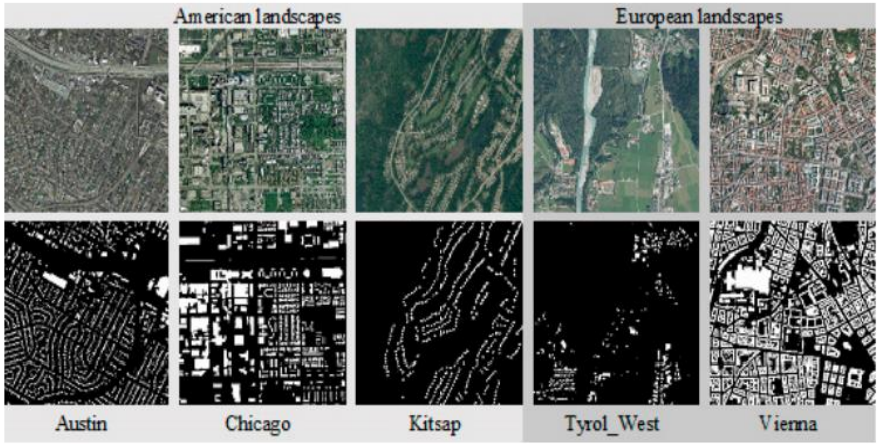
3



ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ ДАННЫХ

В работе использовался Inria Aerial Image Labeling Dataset (https://project.inria.fr/aerialimagelabeling/). Набор данных содержит 180 спутниковых снимков в разрешении 5000x5000 пикселей и 180 масок.

Город	Кол-во изображений	Общая площадь
Остин, Техас	36	81 км^2
Чикаго, Иллинойс	36	81 км^2
Китсап, Вашингтон	36	81 км^2
Вена, Австрия	36	81 км^2
Западный Тироль, Австрия	36	81 км^2
Всего	180	405 км^2

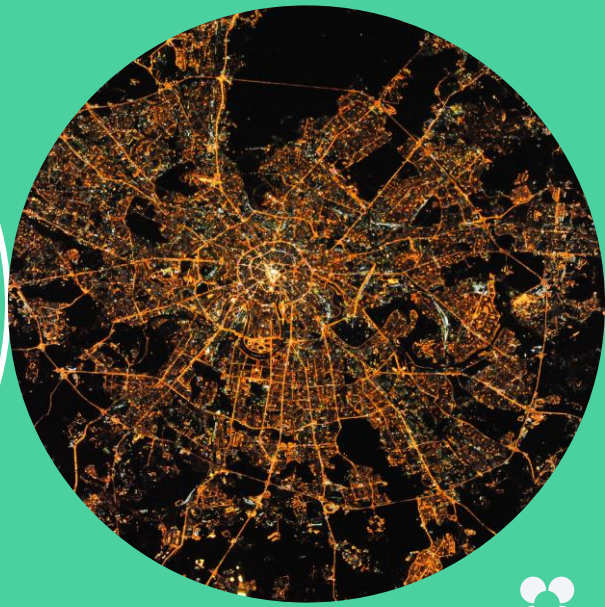


ПРЕДОБРАБОТКА ДАННЫХ

Было принято решение уменьшить размер изображений в тренировочном наборе данных. Сначала это было сделано путем понижения разрешения до 512*512 (в 10!!! раз, не самое удачное решение).
Далее из каждой картинки 5000x5000 из изначального набора данных было вырезано по 4 картинки размером 224x224 пикселей. Всего было получено 720 изображений.
Это позволило снизить время обучения модели и в то же время улучшило результаты.

ПРОЦЕСС МОДЕЛИРОВАНИЯ

4



ЦЕЛЕВЫЕ МЕТРИКИ

Целевой метрикой был выбран коэффициент Дайса (Dice coefficient), который показывает меру сходства — в нашем случае, площадь правильно отмеченных сегментов.

$$Dice = \frac{2 \cdot |mask \cap prediction|}{|mask| + |prediction|}$$

ВЫБРАННЫЕ МОДЕЛИ

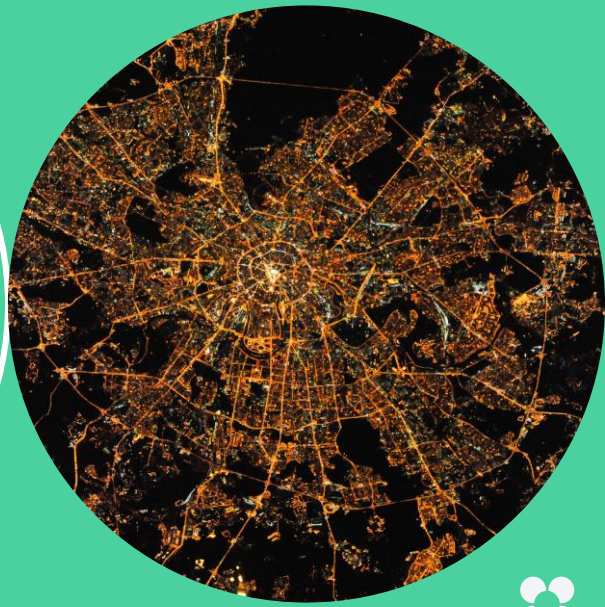
Были использованы разные архитектуры сетей:

- **UNET_VGG** (из Keras)
- **UNET_Resnet34** (из FastAI)
- **UNET_VGG16 (BN)** (из FastAI)
- Модифицированная **TernausNet** (базируется на **UNET** с **VGG16**)

  (https://arxiv.org/pdf/1801.05746.pdf)

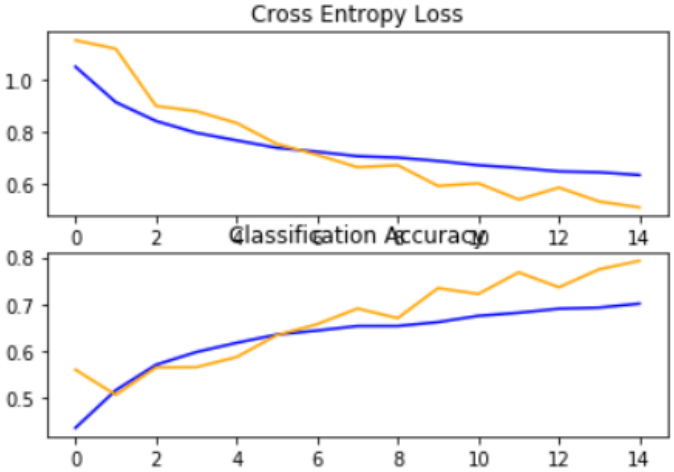
РЕЗУЛЬТАТЫ

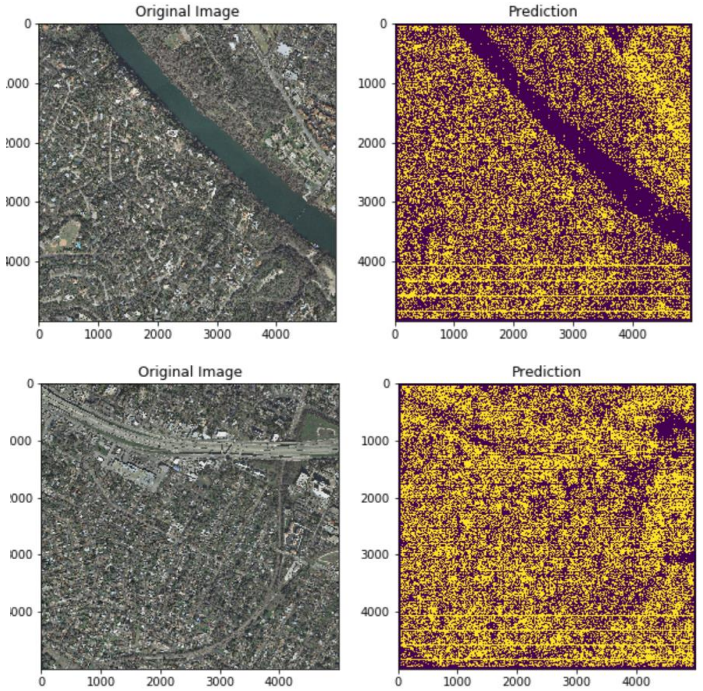
5



KERAS.VGG_UNET (128x128, cropped images)

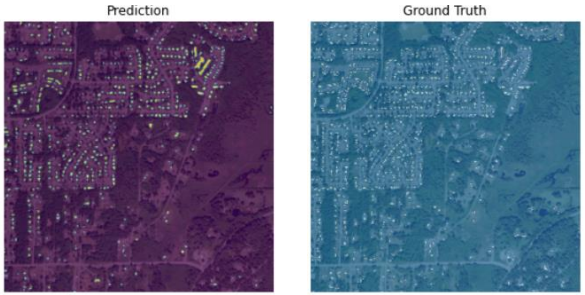
Train_loss	0,632
Val_loss	0,506
Accuracy	0,711

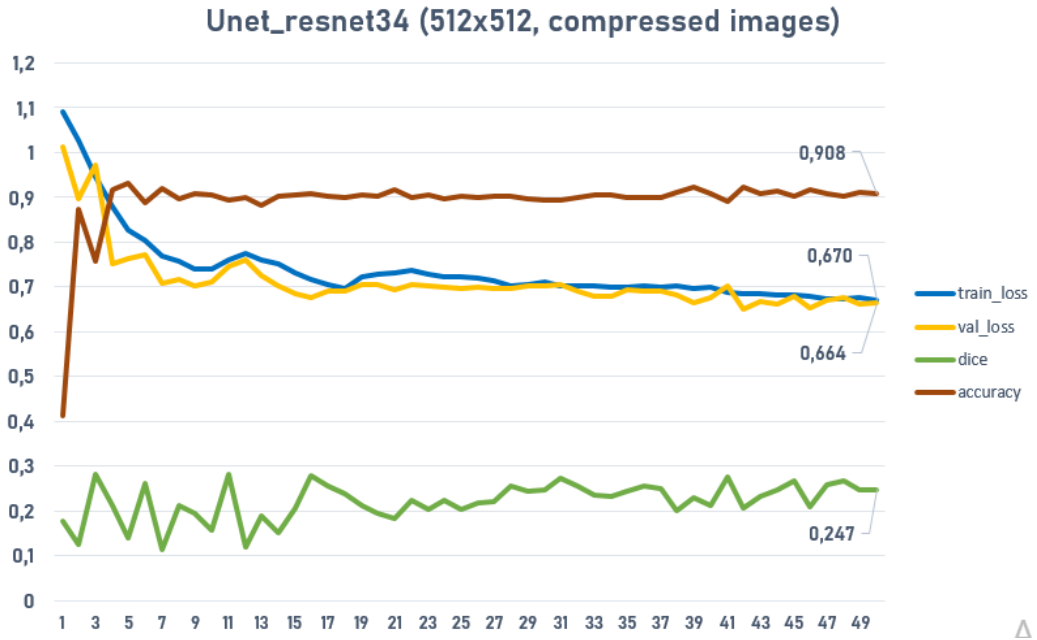




Unet_resnet34 (512x512, compressed images)

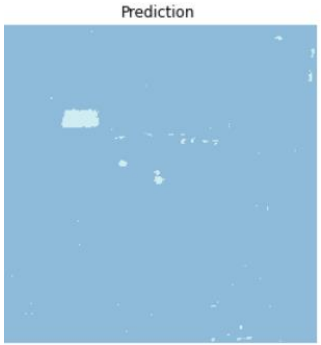
Train_loss	0,665
Val_loss	0,658
Accuracy	0,912
Dice	0,243

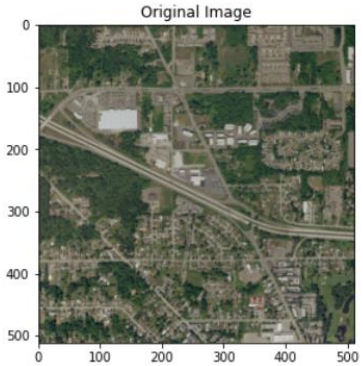


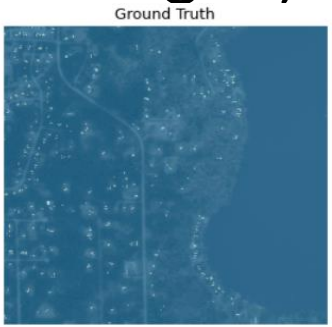


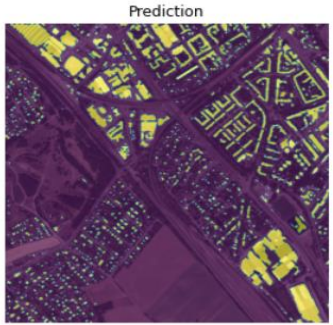
Unet_resnet34 (512x512, compressed images)

Train_loss	**0,665**
Val_loss	**0,658**
Accuracy	**0,912**
Dice	**0,243**



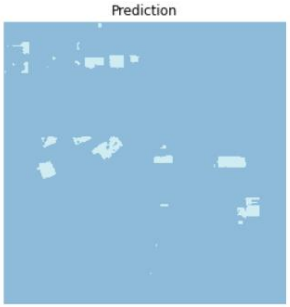


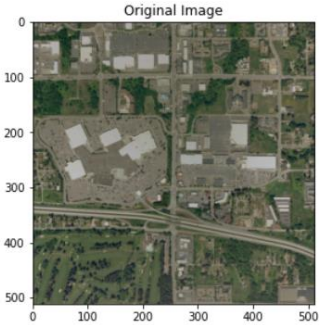


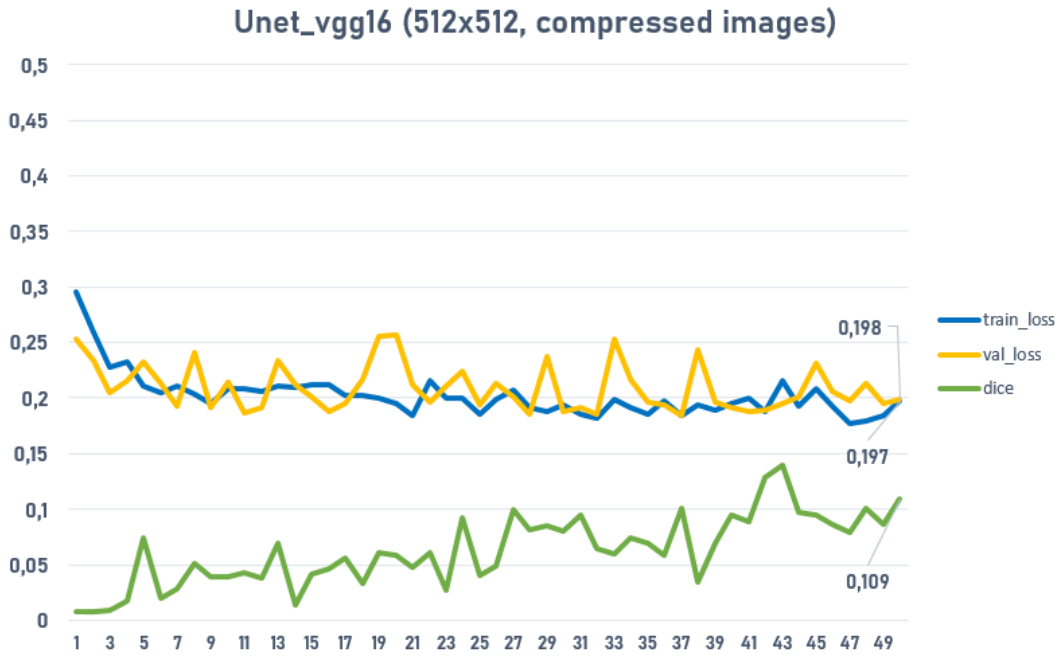


U-net_VGG16 (512x512, compressed images)

Train_loss	**0,197**
Val_loss	**0,198**
Dice	**0,109**

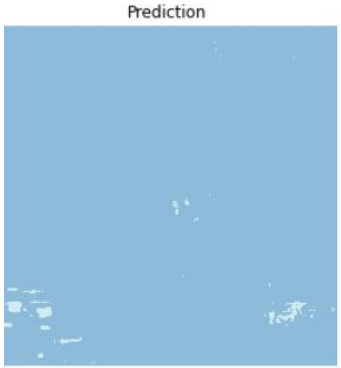


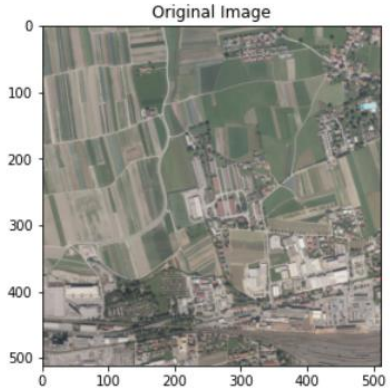




U-net_VGG16 (512x512, compressed images)

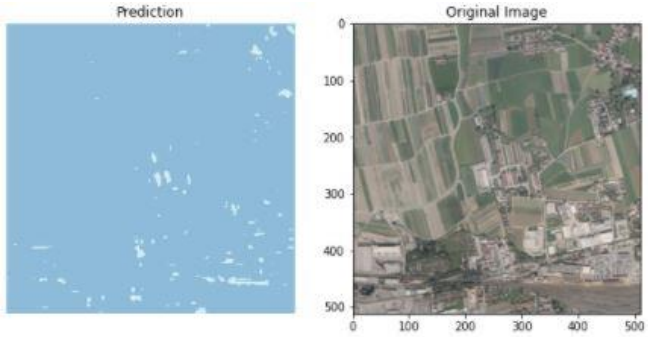
Train_loss	**0,197**
Val_loss	**0,198**
Dice	**0,109**

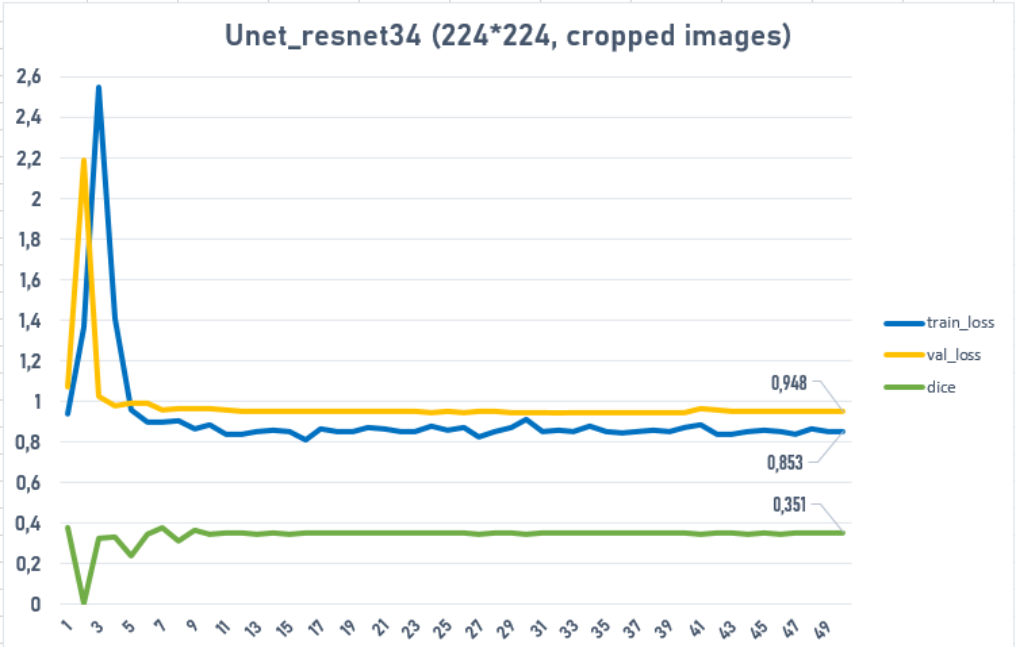




U-net_resnet34 (224x224, cropped images)

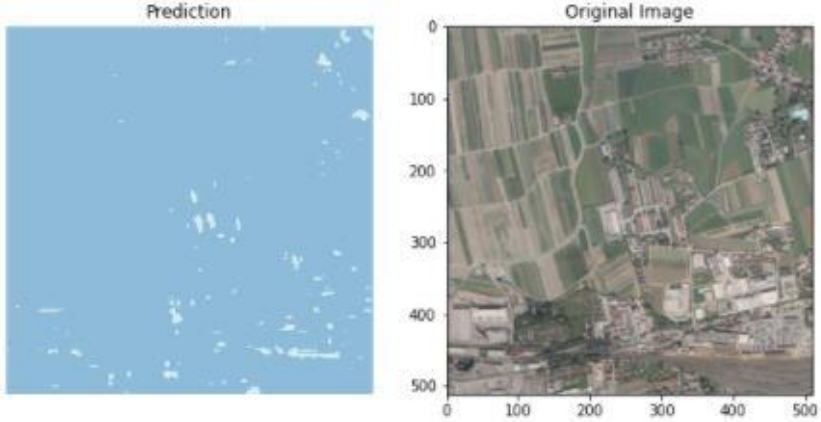
Train_loss	**0,853**
Val_loss	**0,948**
Dice	**0,351**

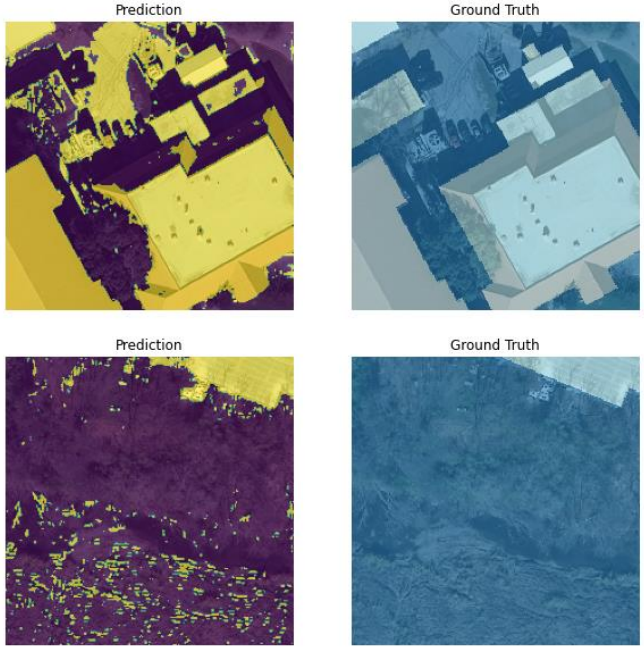




U-net_resnet34 (224x224, cropped images)

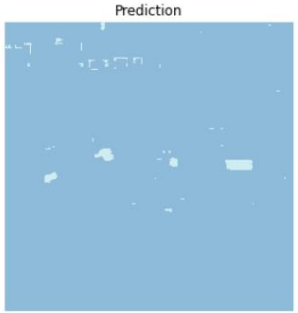
Train_loss	0,853
Val_loss	0,948
Dice	0,351

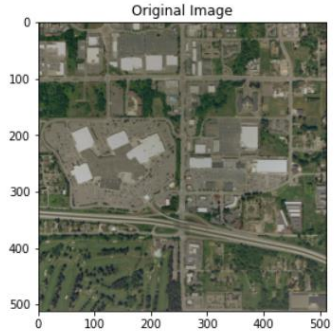


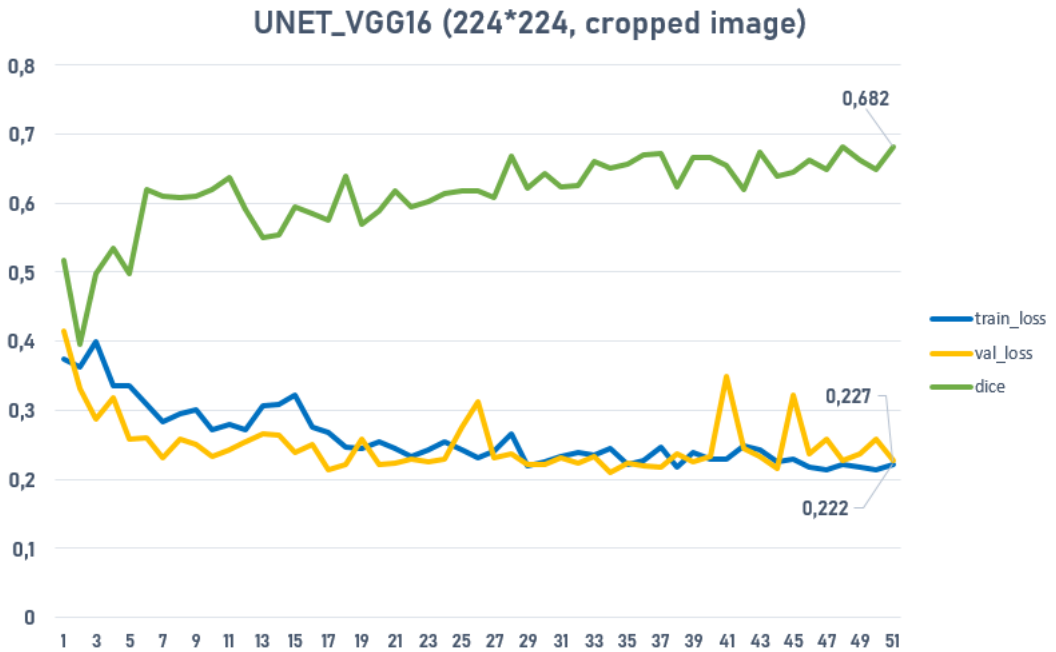


U-net_VGG16 (224x224, cropped images)

Train_loss	**0,222**
Val_loss	**0,227**
Dice	**0,682**

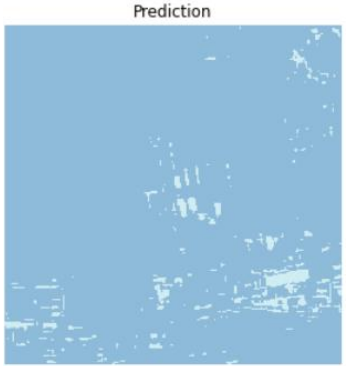


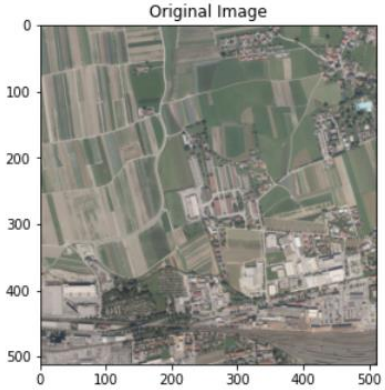


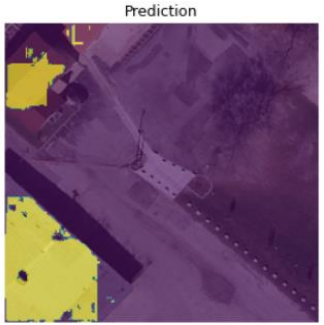


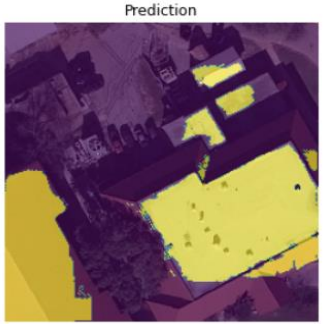
U-net_VGG16 (224x224, cropped images)

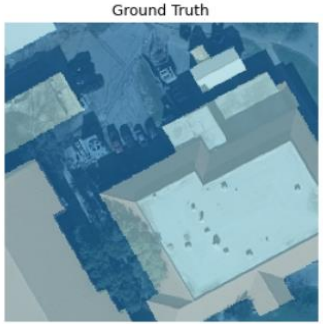
Train_loss	**0,222**
Val_loss	**0,227**
Dice	**0,682**





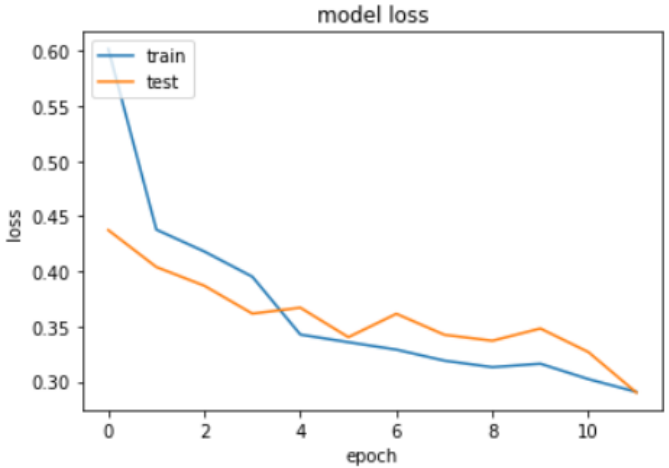






TernausNet * (224x224, cropped images)

Train_loss	0,301
Val_loss	0,298
Dice	0,598

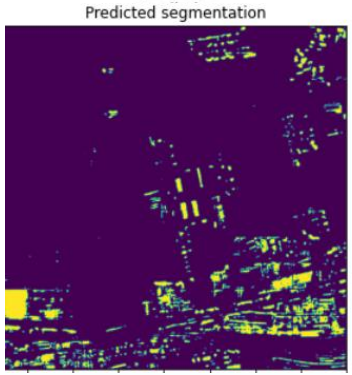


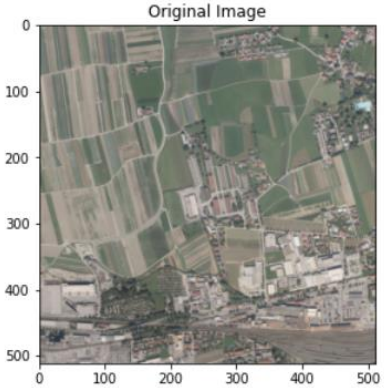

```python
def TernausNet(self):
    # ============================================================================
    # Encoder part
    # ============================================================================
    i   = Input((self.input_size,self.input_size,3))
    c1  = Conv2D(64,(3,3),activation='relu',padding='same')(i)
    c2  = Conv2D(64,(3,3),activation='relu',padding='same')(c1)
    mp1 = MaxPooling2D((2,2))(c2)
    c3  = Conv2D(128,(3,3),activation='relu',padding='same')(mp1)
    mp2 = MaxPooling2D((2,2))(c3)
    # ============================================================================
    # Center part
    # ============================================================================
    c4 = Conv2D(128,(3,3),activation='relu',padding='same')(mp2)
    # ============================================================================
    # Decoder part
    # ============================================================================
    d1 = Conv2DTranspose(64,(3,3),activation='relu',padding='same',strides=2)(c4)
    m1 = concatenate([d1,c3])
    d2 = Conv2DTranspose(32,(3,3),activation='relu',padding='same',strides=2)(m1)
    m2 = concatenate([d2,c2])
    c5 = Conv2D(64,(3,3),activation='relu',padding='same')(m2)
    c6 = Conv2D(1,(3,3),activation='sigmoid',padding='same')(c5)

    model = Model(inputs=i,outputs=c6)

    return model
```

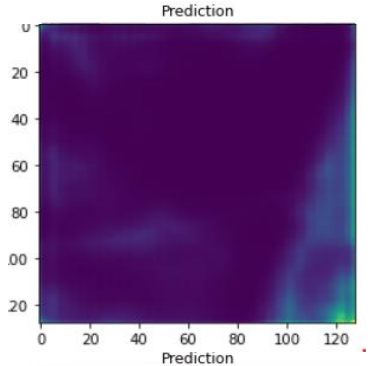
* https://arxiv.org/pdf/1801.05746.pdf

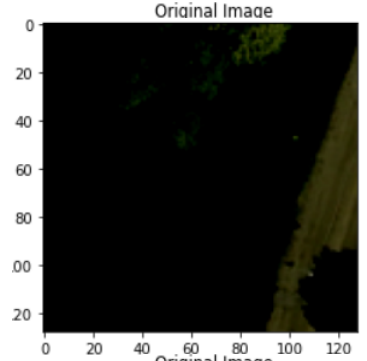
TernausNet (224x224, cropped images)

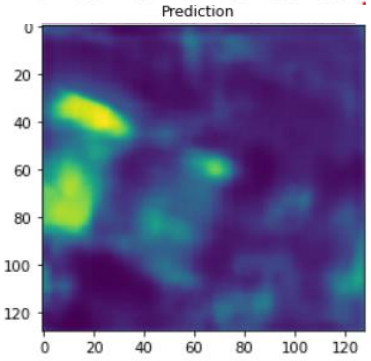
Train_loss	**0,301**
Val_loss	**0,298**
Dice	**0,598**

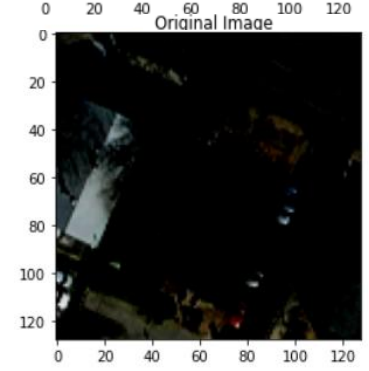




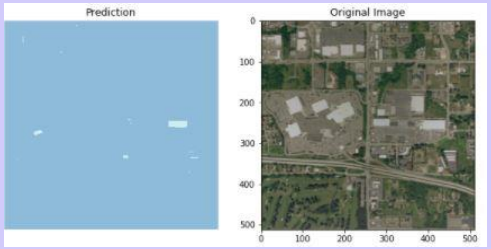
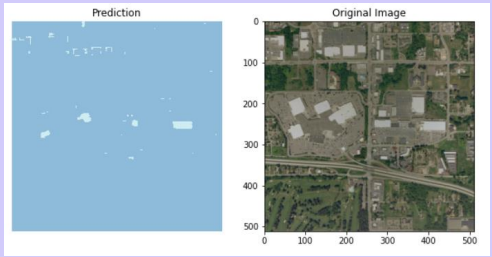
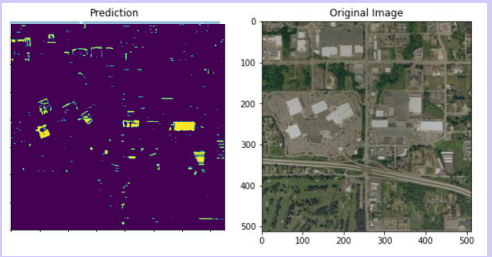
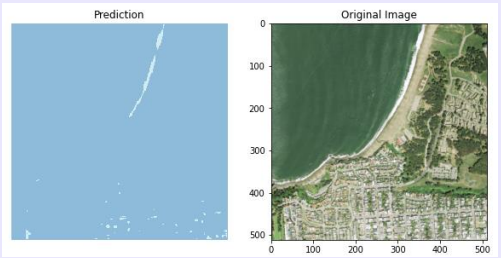
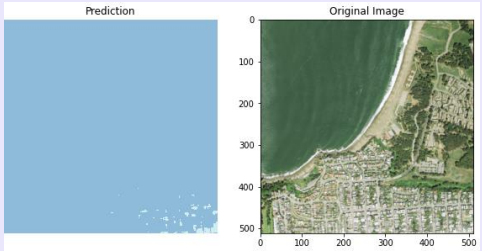
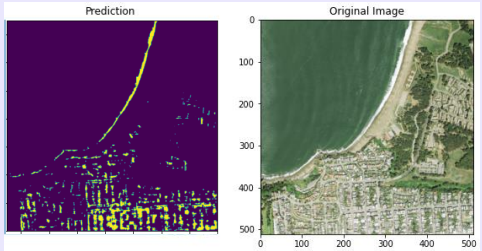






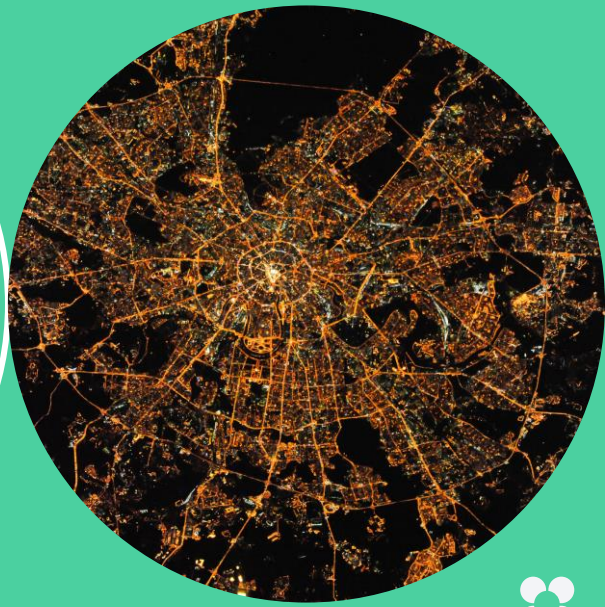


Сравнение результатов

U-net_resnet34	U-net_VGG16	TernausNet
		
		

ВЫВОДЫ

6



ВЫВОДЫ

Было разработано несколько моделей, решающих задачу семантической сегментации спутниковых снимков. У лучших моделей коэффициент Дайса приближался к 0.7.

Качественная оценка сегментации, производимой моделями из данной работы, позволяет считать их применимыми для разметки спутниковых снимков.

КУДА ДАЛЬШЕ?

Далее сегментацию можно расширить на другие объекты на спутниковых снимках: дороги, леса, поля, реки.

Спасибо за внимание!