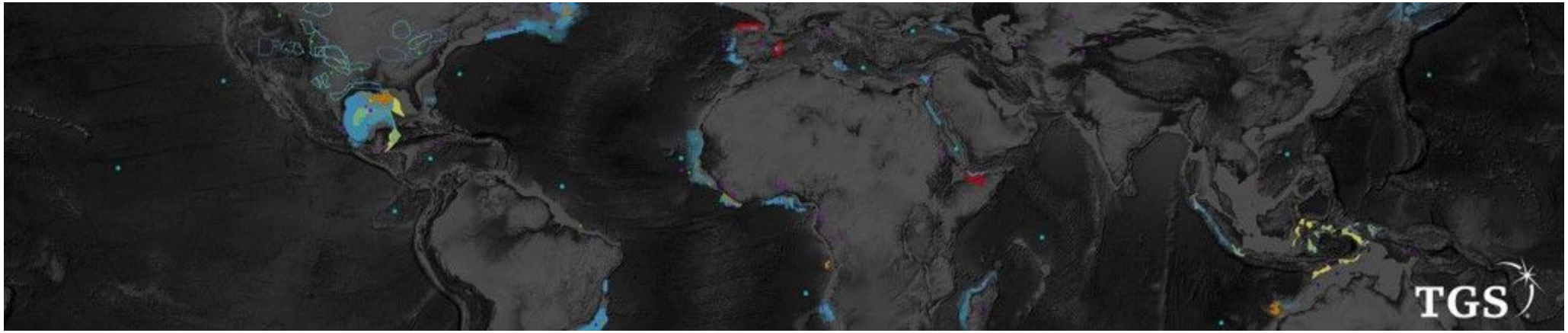
Анализ подходов решений некоторых практических задач семантической сегментации

Чернышев Вадим, гр. 320
Научный руководитель: к.ф.-м.н. Конушин Антон Сергеевич

Сами соревнования

TGS Salt Identification Challenge



Carvana Image Masking Challenge

Carvana

Особенности задачи:

- Крупное разрешение(1918x1280)

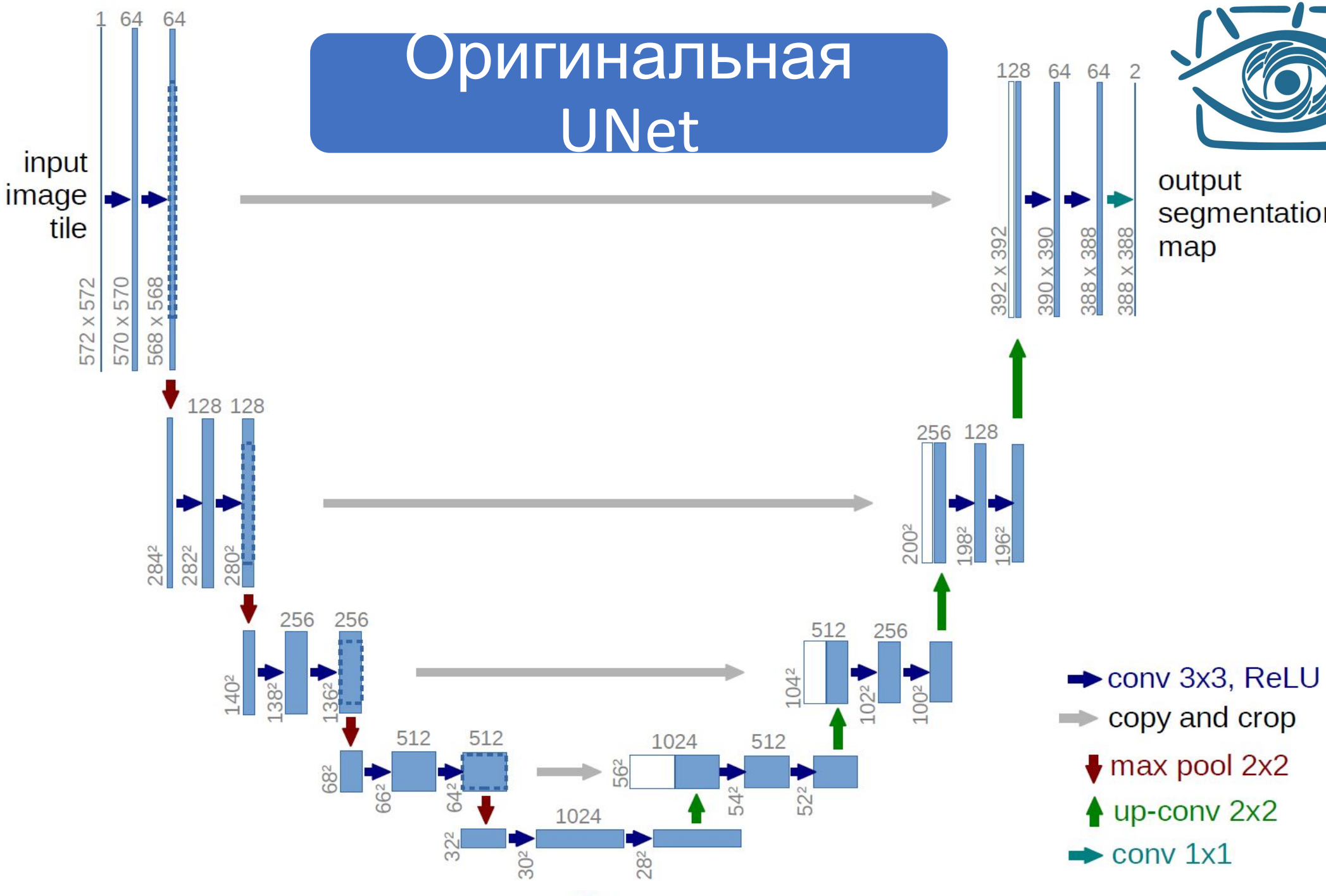
- Важность каждого пикселя
  (2.5 пикселя в итоге отделяют 1ое и 2ое место)

- Большая выборка данных

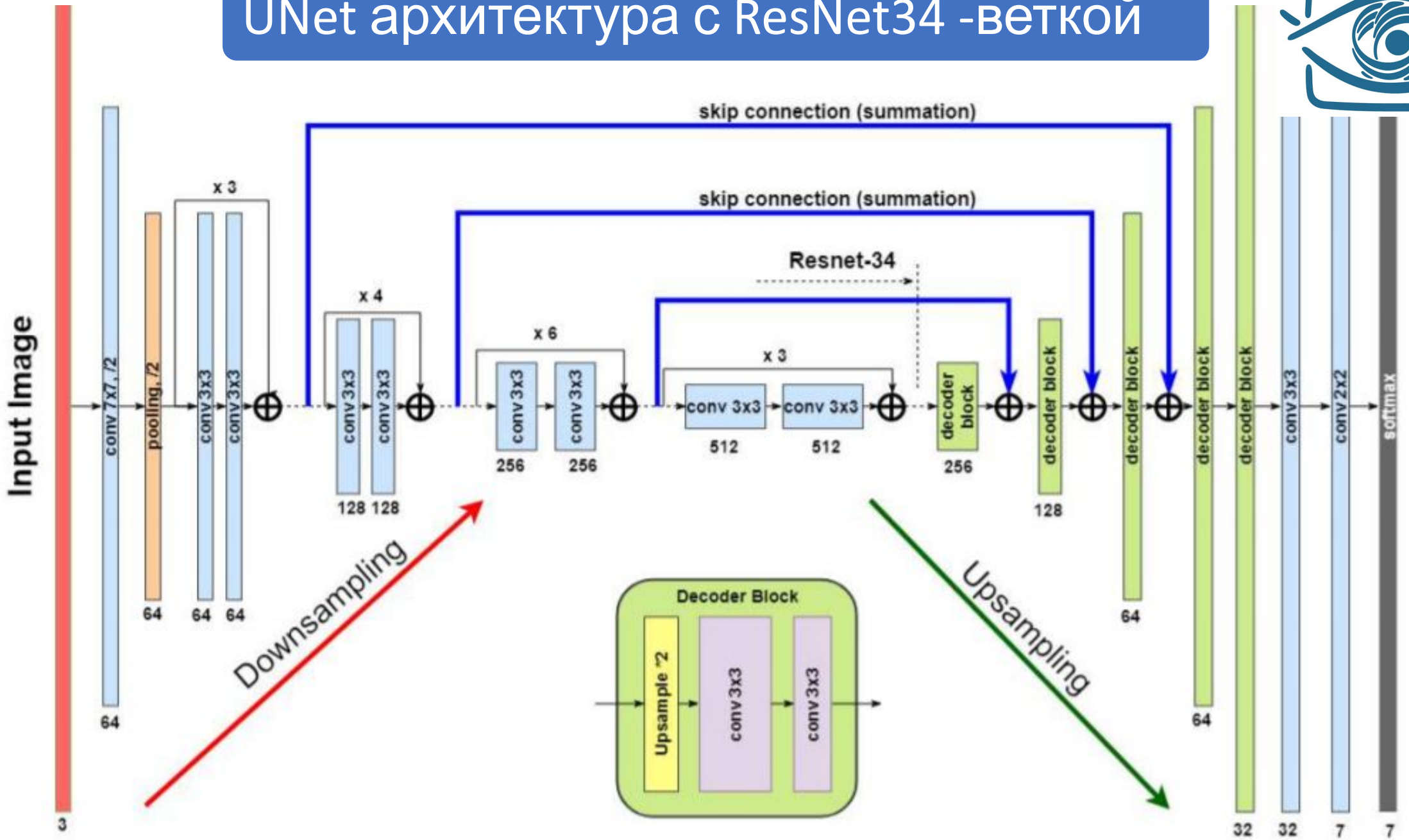
https://www.kaggle.com/c/carvana-image-masking-challenge/overview


CARVANA

Оригинальная UNet



UNet архитектура с ResNet34 -веткой

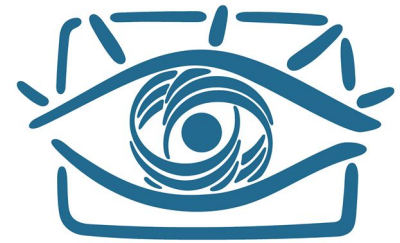


Архитектуры победителей: Первый участник

- UNet-подобная сеть с ResNet-34
- Предобучение на ImageNet – хорошо
- Optimizer – Adam
- Циклический темп обучения(после 30-ой эпохи)
  (Цикл(1e-6, 1e-5, 1e-4, 1e-5, 1e-6) по 2 эпохи в цикле)
- Loss = BSE - Ln(DICE)

$$BCE = -\sum_i \left(y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i)\right)$$

$$DICE = 2\frac{\sum_i y_i p_i}{\sum_u y_i + \sum p_i}$$



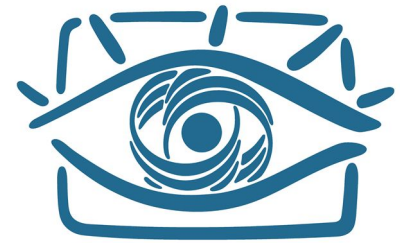
Архитектуры победителей: Первый участник

fold	Public Score	Public LB Place	Private Score	Private LB Place
0	0.99722	7	0.997166	7
1	0.997203	8	0.997196	6
2	0.997216	8	0.99711	12
3	0.997228	6	0.99718	7
4	0.997243	5	0.997134	9
mean	0.997277	3	0.997228	5

Второй участник:
Быстрее, чем VGG11

- LinkNet
- Coder - легковесная ResNet34, предобученая на ImageNet
- Мягкая аугментация – горизонтальные перевороты, 10% масштабирование, 100-пиксельные сдвиги, повороты на 5°
- Optimizer - Adam (или RMSProp)
- Темп обучения: 1e-4 (12 эпох) 1e-5 (6 эпох)
- Loss = 1 + BSE - DICE
- TTA - горизонтальные перевороты

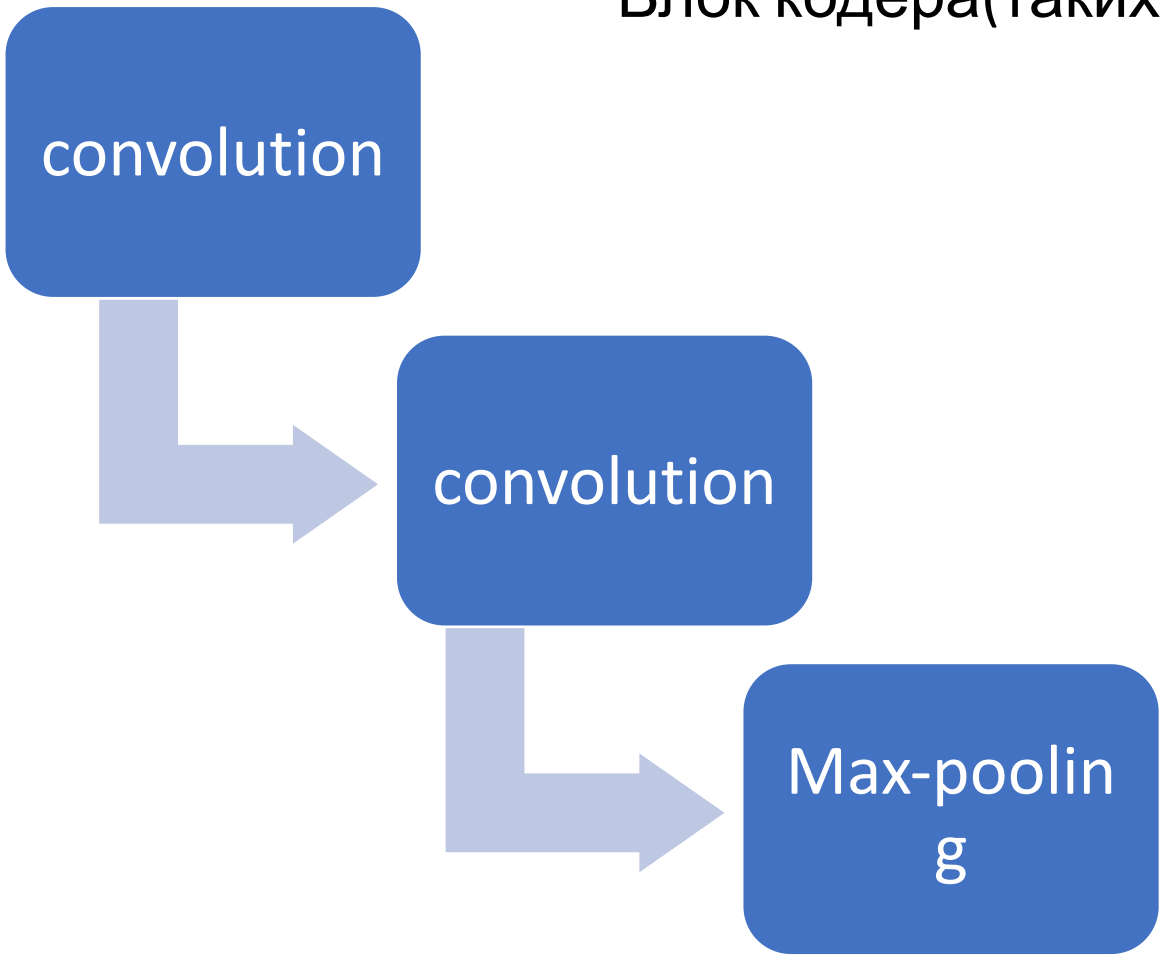
Второй участник: Быстрее, чем Unet



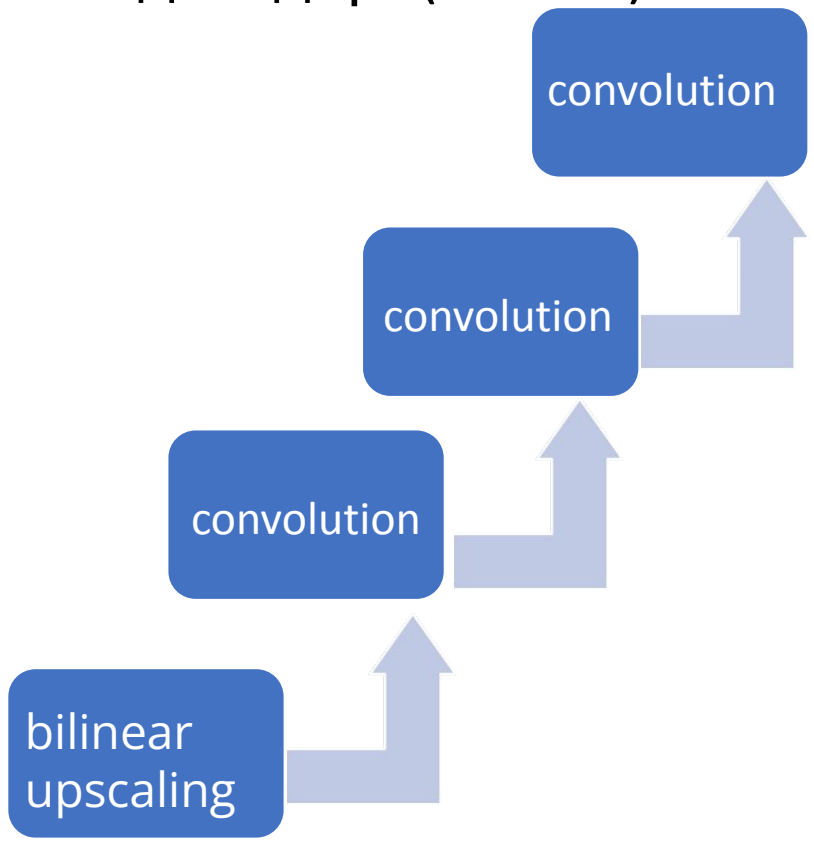
folds	fold	dice	mean	folds	fold	dice	mean	folds	fold	dice	mean
f04a_s55_h_clahe	0	0.997170		fma_s44_cycle_(	0	0.997156		cycle ens	0	0.997253	
	1	0.997078			1	0.997139			1	0.997240	
	2	0.997182			2	0.997093			2	0.997199	
	3	0.997071			3	0.996963			3	0.997091	
	4	0.996997	0.997100		4	0.997113	0.997093		4	0.997223	0.997201
f04a rmsprop	0	0.997165		fma_s55_cycle_(	0	0.997144					
	1	0.997073			1	0.997136					
	2	0.997189			2	0.997068					
	3	0.997109			3	0.996944					
	4	0.997014	0.997110		4	0.997135	0.997085				
f04a_cb	0	0.997166		fma_s88_cycle_(	0	0.997161					
	1	0.997041			1	0.997146					
	2	0.997185			2	0.997069					
	3	0.997072			3	0.996952					
	4	0.996998	0.997092		4	0.997129	0.997091				

Третий участник

Блок кодера(таких 6)



Блок декодера(тоже 6)



Третий участник

	UNet с нуля	UNet-VGG-11
Модель:	- кастомная UNet	- UNet на основе VGG-11
Вес а:	- случайные	- pretrained VGG-11(ImageNet)
Folds:	- 7(случайно)	- 7 (разбиты по S масок)
Число эпох:	- 250	- 60
Темп обучения:	- lr*0.5 за 100 эпох	- (10 – lr), (5 - lr *0.1), (5 - lr *0.01)
Аугментация:	- тяжелая	- тяжелая(15 эпох) и легкая(45)

Третий участник

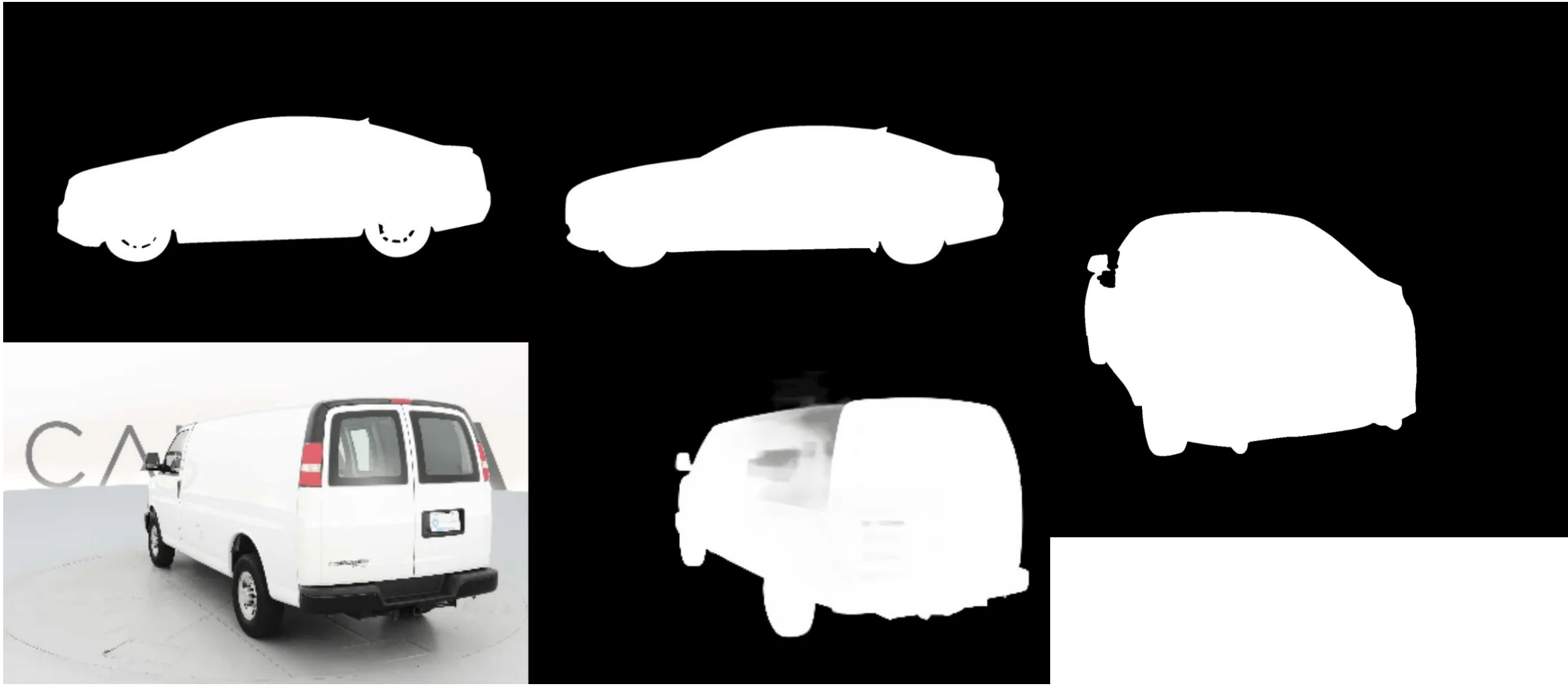
- **Аугментация**:
  - 1) **тяжелая**: масштабирование
    изменение яркости
    изменение насыщенности
    преобразование в оттенки серого
    изменение контраста
    вращение
  - 2) **Легкая**: масштабирование и вращение
- **Вход** - 1024x1024
- **Loss** =1+BSE – DICE
- Коэффициенты BSE взяты с весами, растущими у границы масок

Третий участник

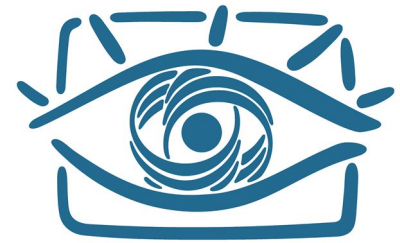
Unet from scratch				**Unet-VGG-11**		
Fold	CV dice score	Public LB		Fold	CV dice score	Public LB
0	0.9970415131	0.997065		0	0.9970812679	
1	0.9969061218	0.997009		1	0.9971920427	
2	0.9971917226	0.997082		2	0.9970880899	
3	0.9971542991	0.997037		3	0.9970854812	
4	0.9971187972	0.997025		4	0.9971396001	
5	0.9970837553	0.997079		5	0.9971988606	
6	0.9971625434	0.997073		6	0.9970485309	
mean	**0.9970941075**			**mean**	**0.9971191248**	
ensemble	-	**0.997228**		**ensemble**	-	**0.997226**

Результат Ансамбля трех участников -0.99733

Найденные ошибки:



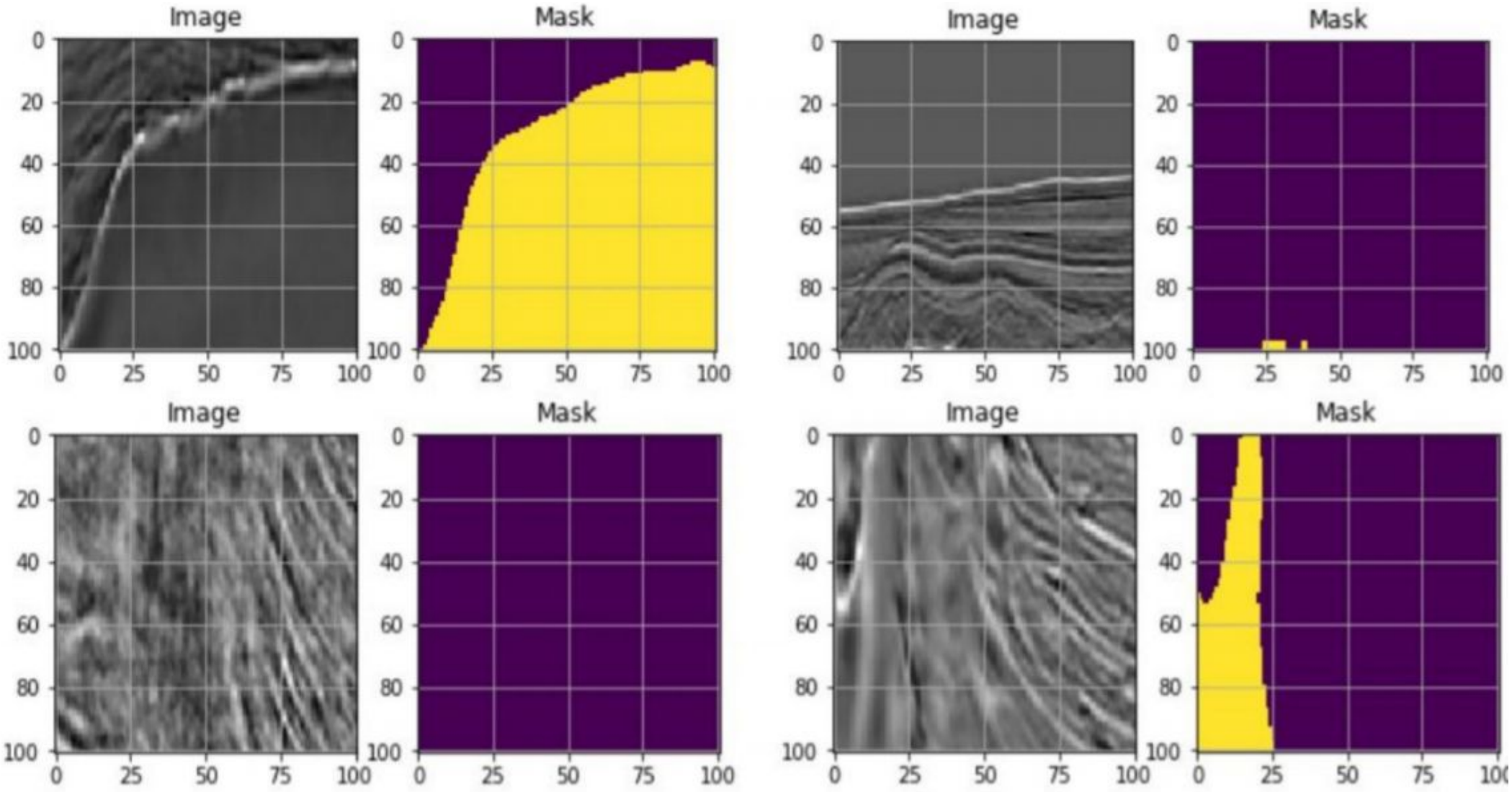
Подходы других участников:



- PSPNet(0.9969) срабатывает хуже
- Псевдолейблинг
- Фильтрация данных
- Обучение некоторых моделей ансамбля на поиск фона, либо
  создание псевдоклассов
- Практически у всех участников модели схожи

```
Def SoftDICE_loss (y_true, y_pred):
    smooth =1.
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = y_true*y_pred
    score =  (2. * K.sum(intersection) + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)
    return 1. - score
```

TGS Salt detection



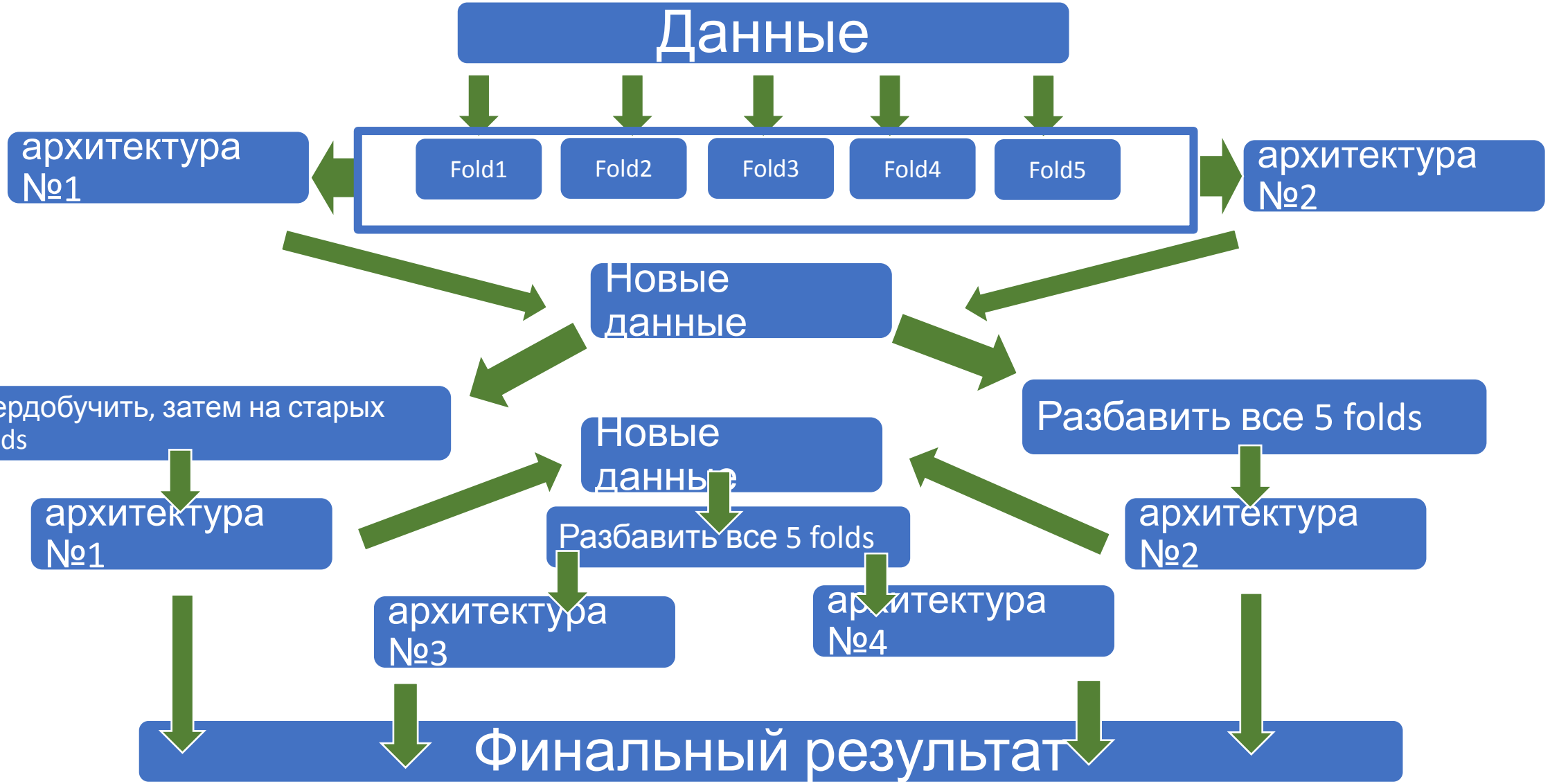
TGS Salt detection

Основные моменты:

- Небольшой размер изображений (примерно 100X100)
- Большое количество пустых данных(с нулевой маской)
- Много не размеченных и мало размеченных картинок
- Для каждой картинки дополнительно дана глубина съёмки

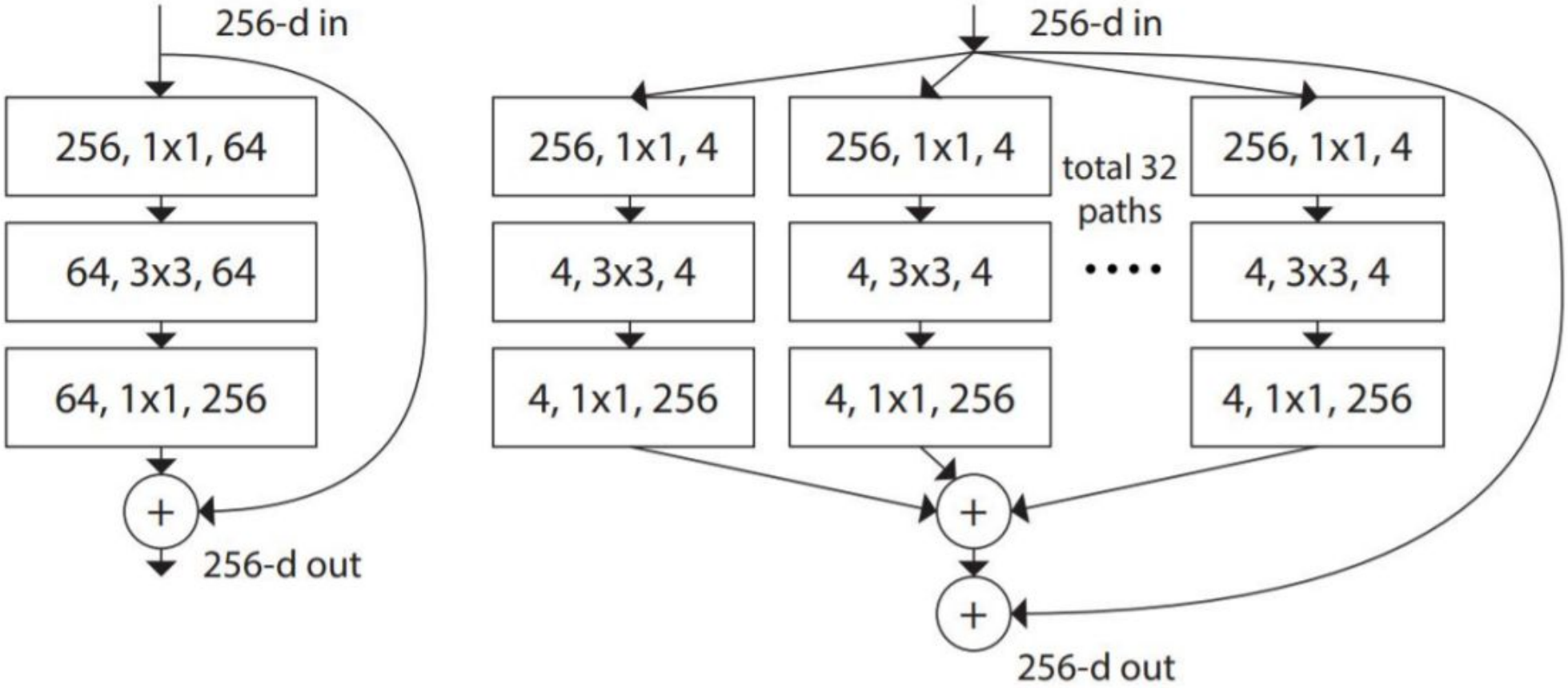
https://www.kaggle.com/c/tgs-salt-identification-challenge/overview

Разбор лучшего решения



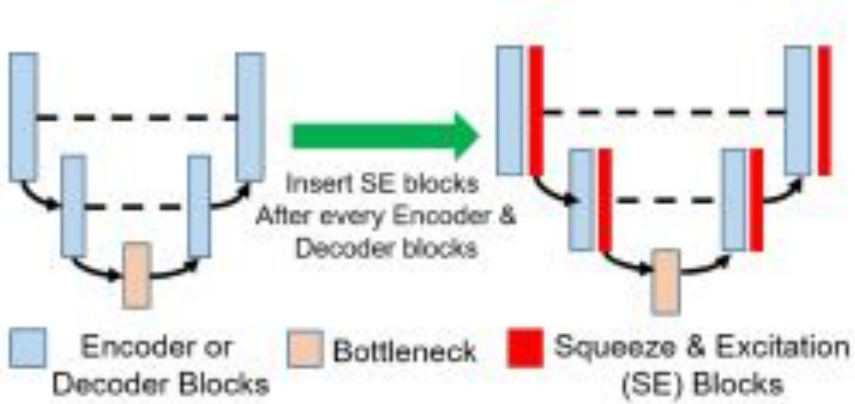
ResNet блок

ResNext блок

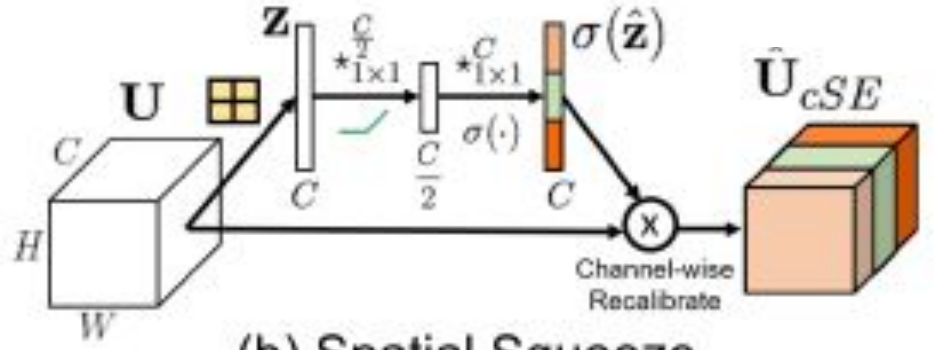


https://arxiv.org/abs/1611.05431

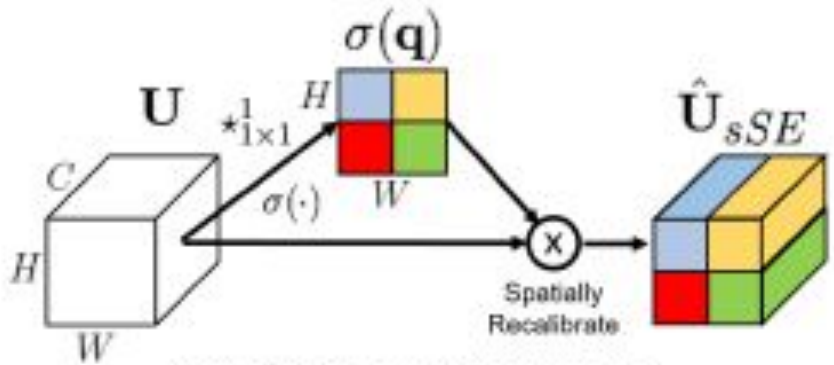
Spatial and Channel Squeeze & Excitation



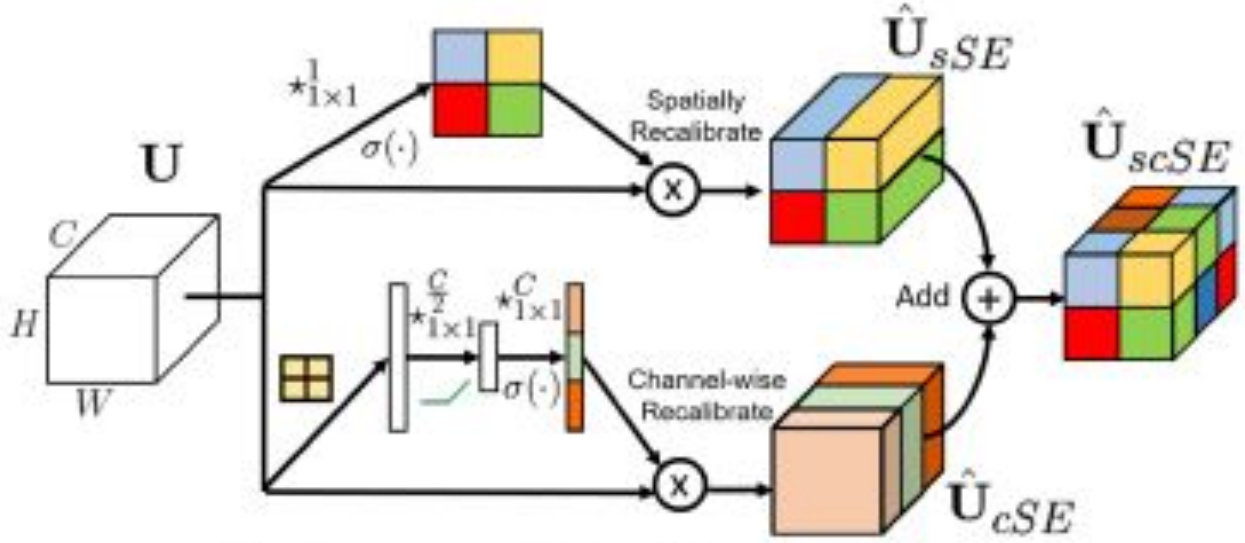
(a) SE block in F-CNN



(b) Spatial Squeeze
and Channel Excitation (cSE)

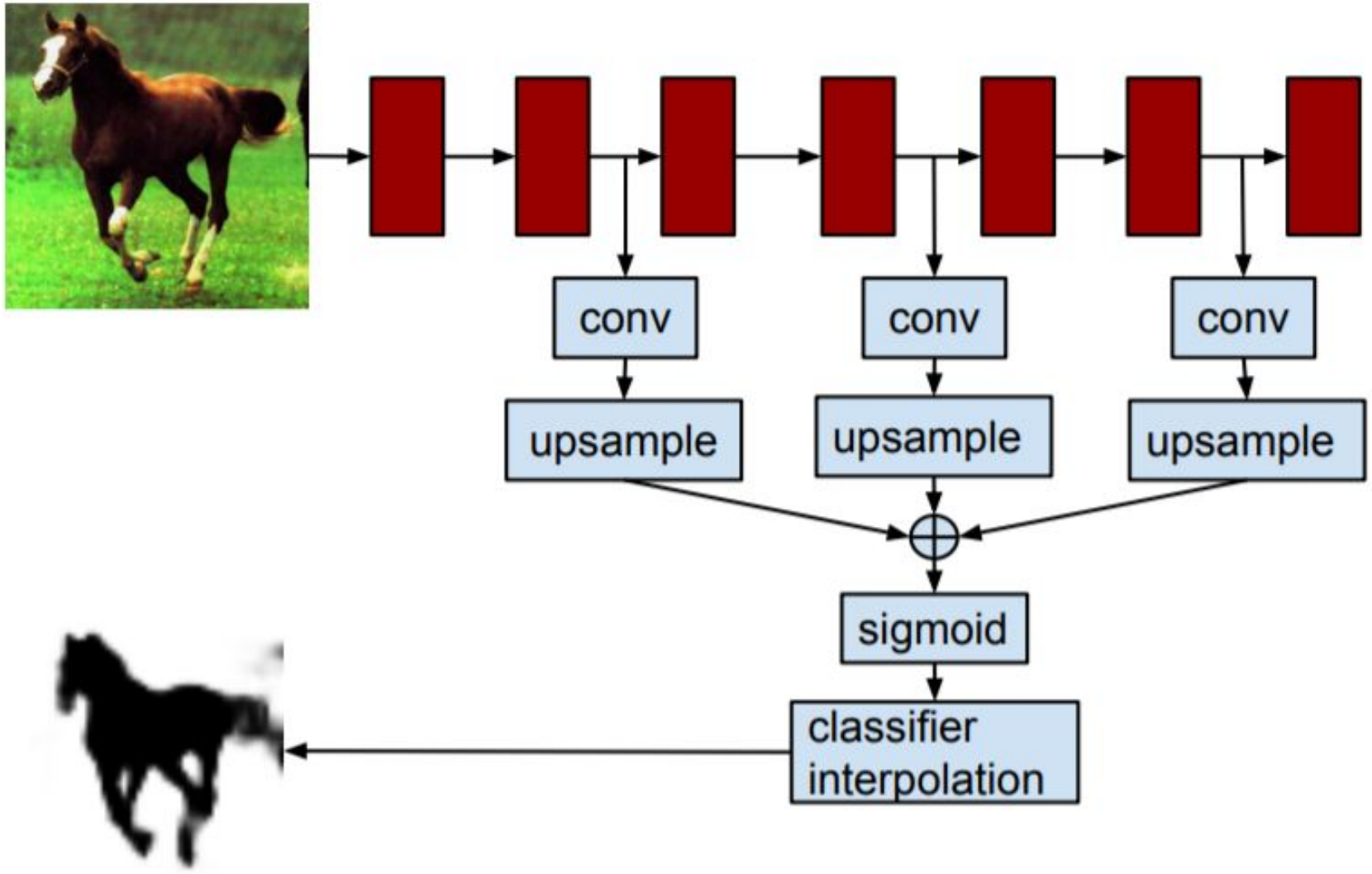


(c) Channel Squeeze
and Spatial Excitation (sSE)



https://arxiv.org/abs/1803.02579

hyper columns



https://arxiv.org/abs/1611.05431

	ResNeXt34	ResNet34		
Вход:	101 -> resize to 192 -> pad to 224	101->resize to 202->pad to 256	101 -> pad to 128	101 -> resize to 128
Mod:	- conv7x7 -> conv3x3 и удален 1-ый max pooling		удален 1-ый max pooling	
decoder:	conv3x3 + BN, Upsampling, scSE	conv3x3, transposed conv	scSE + hyper columns	transposed conv
Losses:	1)BCE+1-Dice. Lr from 0.0001 2)Lovasz(from 0.00005) 3)Lovasz. 4 snapshots with cosine annealing LR, 80 epochs each, LR starting from 0.0001	Lovasz	Lovasz	Lovasz
Optimizer:	RMSprop	SGD Adam	Adam	

Augmentations: HorizontalFlip(p=0.5) ShiftScaleRotate(shift_limit=0.1625, scale_limit=0.6, rotate_limit=0, p=0.7)
RandomBrightness(p=0.2, limit=0.2) RandomContrast(p=0.1, limit=0.2)

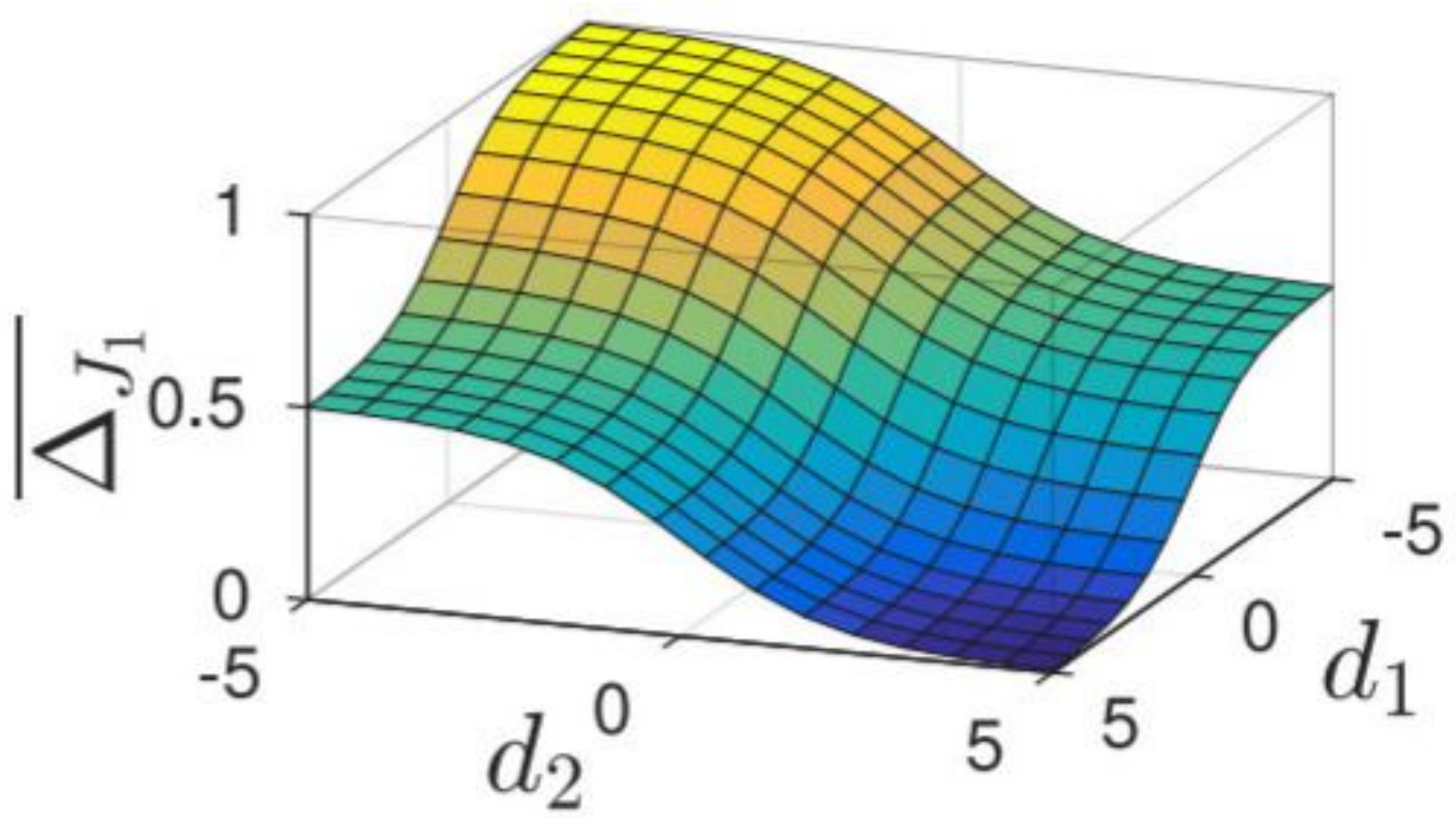
архитектура №1

архитектура №2

архитектура №3

архитектура №4

Функция потерь Lovasz



https://arxiv.org/abs/1705.08790

AlBumentation

- Быстрая аугментация на основе высокооптимизированной библиотеки OpenCV.
- Супер простой, но мощный интерфейс для различных задач
- Легко настроить

https://albumentations.readthedocs.io/en/latest/

применение

```
def my_aug(p=0.5):
    return Compose([
        RandomRotate90(),
        Flip(),

        OneOf([
            IAAAdditiveGaussianNoise(),
            GaussNoise(),
        ], p=0.2),

        OneOf([
            MotionBlur(p=0.2),
            MedianBlur(blur_limit=3, p=0.1),
            Blur(blur_limit=3, p=0.1),],   p=0.2),

        OneOf([
            ChannelShuffle(),
            RGBShift(),
            RandomBrightnessContrast(),],  p=0.3),
    ], p=p)
```

Примеры


CLAHE
RandomContrast
RandomGamma
RandomBrightness
Original image
RGBShift
Blur
MedianBlur
ToGray
JpegCompression
HueSaturationValue
ChannelShuffle

AlBumentation использовали:

- Carvana Image Masking Challenge (1st place)
- Data Science Bowl 2018(1)
- Humpback Whale Identification(5)
- TGS Salt Identification Challenge(1)
- APTOS 2019 Blindness Detection(1)
- SIIM-ACR Pneumothorax Segmentation(7)
- iMaterialist (Fashion) 2019 at FGVC6(1,4)
- Google Landmark Recognition 2019(20)
- Inclusive Images Challenge(3)
- Neptune - Facial Detection Marathon Match 2019(2)
- Neptune - Facial Re-Identification Marathon Match(2)
- .....

https://albumentations.readthedocs.io/en/latest/hall_of_fame.html

Выводы:

- Использование предобученных сетей в качестве encoder’а –хороший старт
- Albumentation –отличная вещь, для облегчения аугментации
- scSE, hyper columns, псевдолейблинг, циклический lr, искусственные классы, ансабль