

---









# Self-training with Noisy Student improves ImageNet classification

Кульпин Павел БПМИ193

# Введение



- Задача классификации на ImageNet
- Semi-supervised learning
- Цель - улучшить имеющиеся результаты с точки зрения качества

			
<b>mite</b>	<b>container ship</b>	<b>motor scooter</b>	<b>leopard</b>
<div><div></div><div>mite</div><div>black widow</div><div>cockroach</div><div>tick</div><div>starfish</div></div>	<div><div></div><div>container ship</div><div>lifeboat</div><div>amphibian</div><div>fireboat</div><div>drilling platform</div></div>	<div><div></div><div>motor scooter</div><div>go-kart</div><div>moped</div><div>bumper car</div><div>golfcart</div></div>	<div><div></div><div>leopard</div><div>jaguar</div><div>cheetah</div><div>snow leopard</div><div>Egyptian cat</div></div>
			
<b>grille</b>	<b>mushroom</b>	<b>cherry</b>	<b>Madagascar cat</b>
<div><div></div><div>convertible</div><div>grille</div><div>pickup</div><div>beach wagon</div><div>fire engine</div></div>	<div><div></div><div>agaric</div><div>mushroom</div><div>jelly fungus</div><div>gill fungus</div><div>dead-man's-fingers</div></div>	<div><div></div><div>dalmatian</div><div>grape</div><div>elderberry</div><div>ffordshire bullterrier</div><div>currant</div></div>	<div><div></div><div>squirrel monkey</div><div>spider monkey</div><div>titi</div><div>indri</div><div>howler monkey</div></div>

- 1: Learn teacher model  $\theta_*^t$  which minimizes the cross entropy loss on labeled images

$$\frac{1}{n} \sum_{i=1}^n \ell(y_i, f^{noised}(x_i, \theta^t))$$

1. Обучить учителя на размеченных данных.
2. Сгенерировать с его помощью новые (псевдо-) маркировки.
3. Обучить ученика на скомбинированных данных.
4. Рассмотреть ученика, как учителя. Вернуться ко 2 пункту.

- 2: Use a normal (i.e., not noised) teacher model to generate soft or hard pseudo labels for clean (i.e., not distorted) unlabeled images

$$\tilde{y}_i = f(\tilde{x}_i, \theta_*^t), \forall i = 1, \dots, m$$

- 3: Learn an **equal-or-larger** student model  $\theta_*^s$  which minimizes the cross entropy loss on labeled images and unlabeled images with **noise** added to the student model

$$\frac{1}{n} \sum_{i=1}^n \ell(y_i, f^{noised}(x_i, \theta^s)) + \frac{1}{m} \sum_{i=1}^m \ell(\tilde{y}_i, f^{noised}(\tilde{x}_i, \theta^s))$$

- 4: Iterative training: Use the student as a teacher and go back to step 2.

# Почему это работает?



- Модель ученика всегда НЕ меньше модели учителя

Такой подход позволяет ученикам “разбираться” в данных лучше, чем их учителя.

- Модификация данных

Аугментации, фильтрация

- Шум модели ученика

dropout, stochastic depth

steel arch bridge



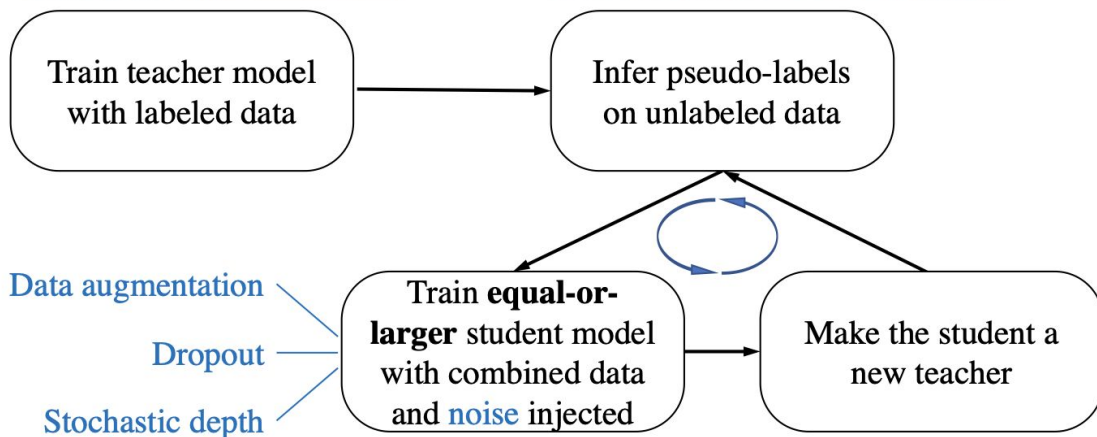
canoe



...



...

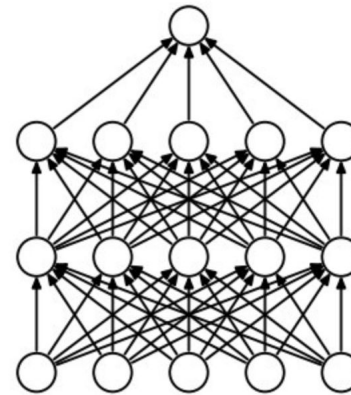


# Шум модели

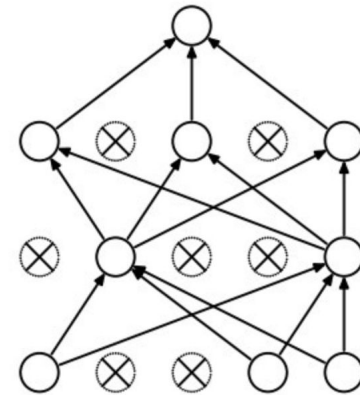


## Dropout

С вероятностью  $p$  отбрасываем нейроны слоя



(a) Standard Neural Net

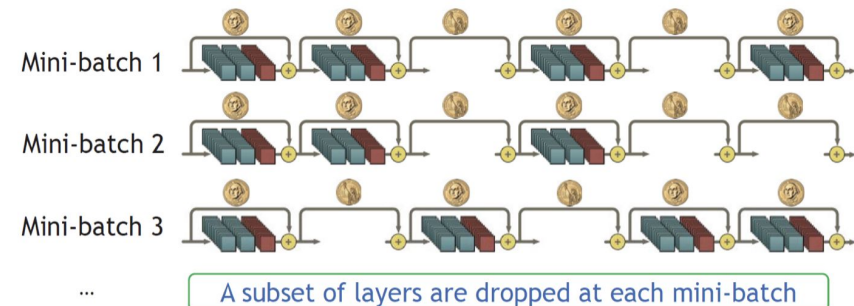


(b) After applying dropout.

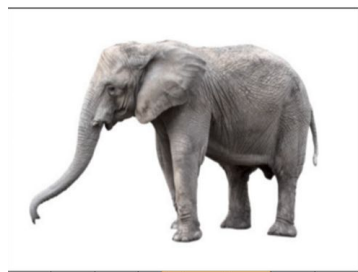
## Stochastic depth

С вероятностью  $p$  отбрасываем не нейроны, а целые слои сети

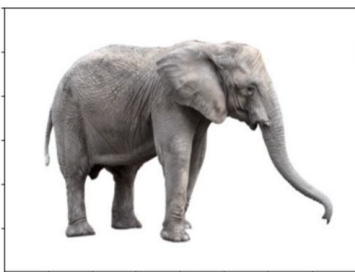
На примере ResNet:



# RandAugment



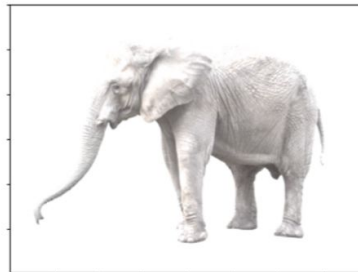
Original Image



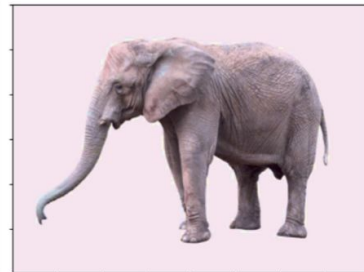
Horizontal Flip



Center Crop Flip



Random Brightness



Channel Dropout



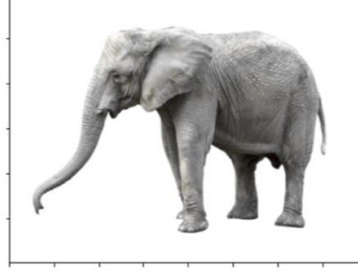
Vertical Flip



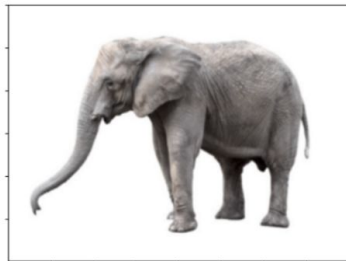
Shift Scale Rotate



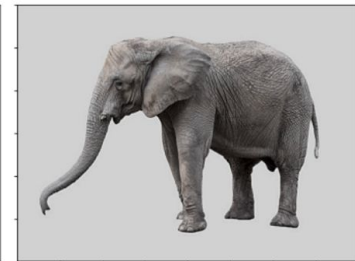
Random Crop



Hue Saturation



Gaussian Blur



Sharpening

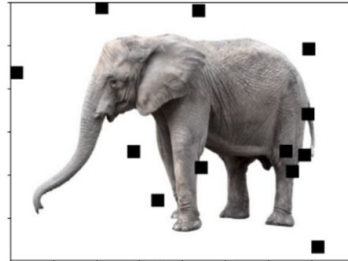
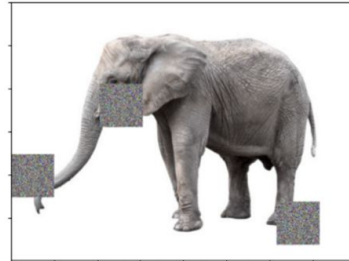


Image Cutout



Cutout + Random noise



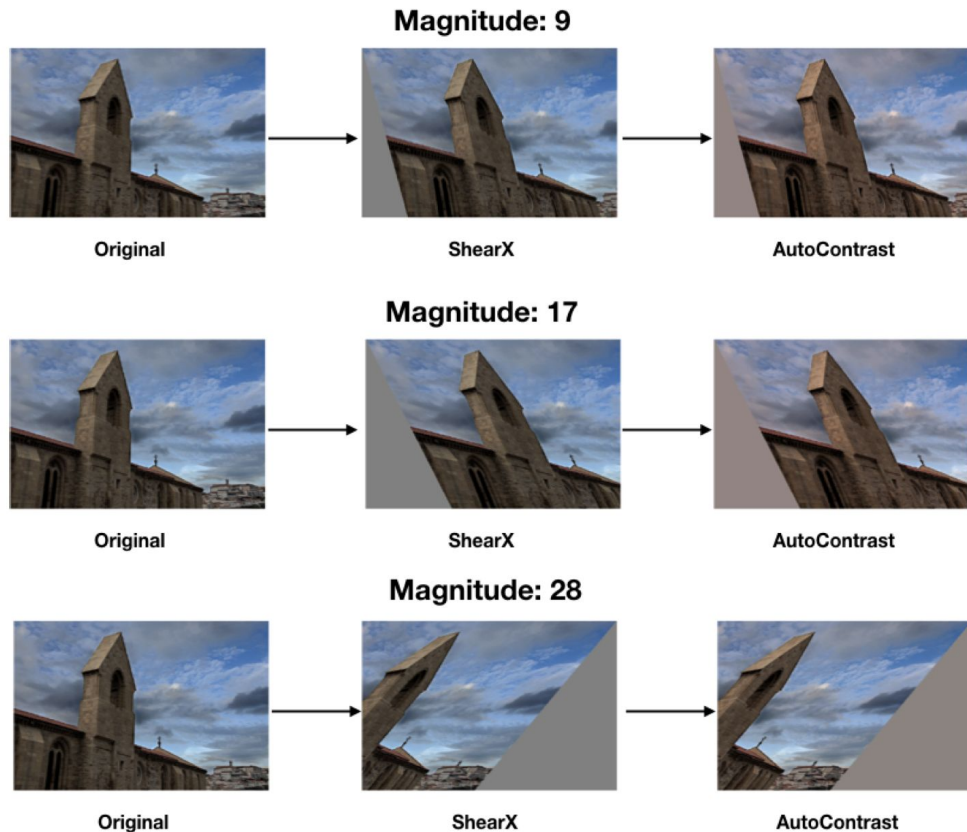
# RandAugment



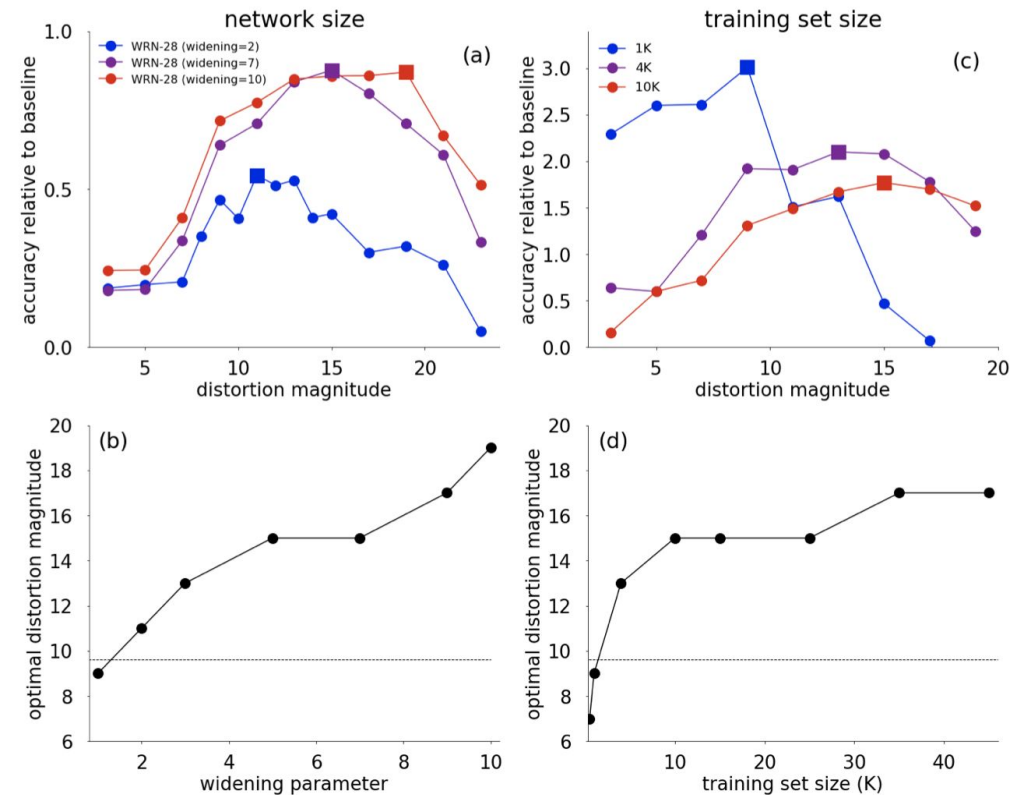
```
transforms = [  
    'Identity', 'AutoContrast', 'Equalize',  
    'Rotate', 'Solarize', 'Color', 'Posterize',  
    'Contrast', 'Brightness', 'Sharpness',  
    'ShearX', 'ShearY', 'TranslateX', 'TranslateY']
```

```
def randaugment(N, M):  
    """Generate a set of distortions.  
  
    Args:  
        N: Number of augmentation transformations to  
            apply sequentially.  
        M: Magnitude for all the transformations.  
    """
```

```
    sampled_ops = np.random.choice(transforms, N)  
    return [(op, M) for op in sampled_ops]
```



# RandAugment



model	augmentation	mAP	search space
ResNet-101	Baseline	38.8	0
	AutoAugment	<b>40.4</b>	$10^{34}$
	RandAugment	40.1	$10^2$
ResNet-200	Baseline	39.9	0
	AutoAugment	<b>42.1</b>	$10^{34}$
	RandAugment	41.9	$10^2$





Еще одна важная отличительная деталь:

Неразмеченные данные могут быть вне класса ImageNet, поэтому снимки, к которым учитель имеет наименьшую степень доверия, отфильтровываются.

Далее количество изображений для каждого класса надо сбалансировать. Где не хватает - дублируем, где слишком много - выбираем изображения с максимальной уверенностью.

Цель создателей - упростить архитектуру нейросетей для работы с изображениями и уменьшить время работы.

Идея - взять простую, но хорошо работающую сеть и оптимально ее масштабировать.

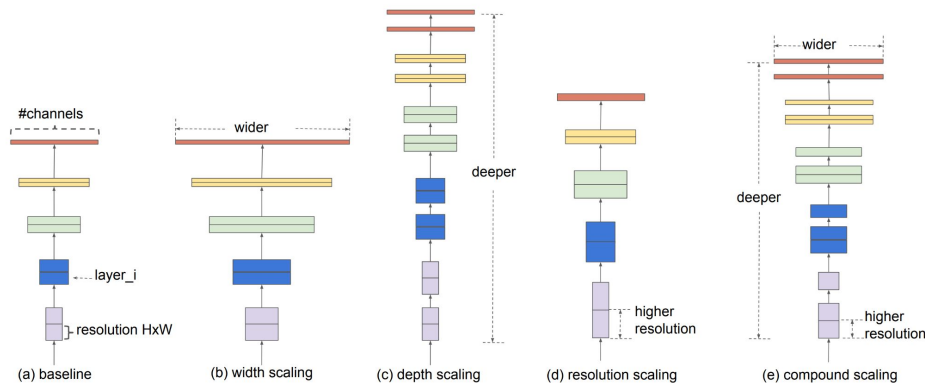
$$depth : d = \alpha^\phi$$

$$width : w = \beta^\phi$$

$$resolution : r = \gamma^\phi$$

$$s.t. \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

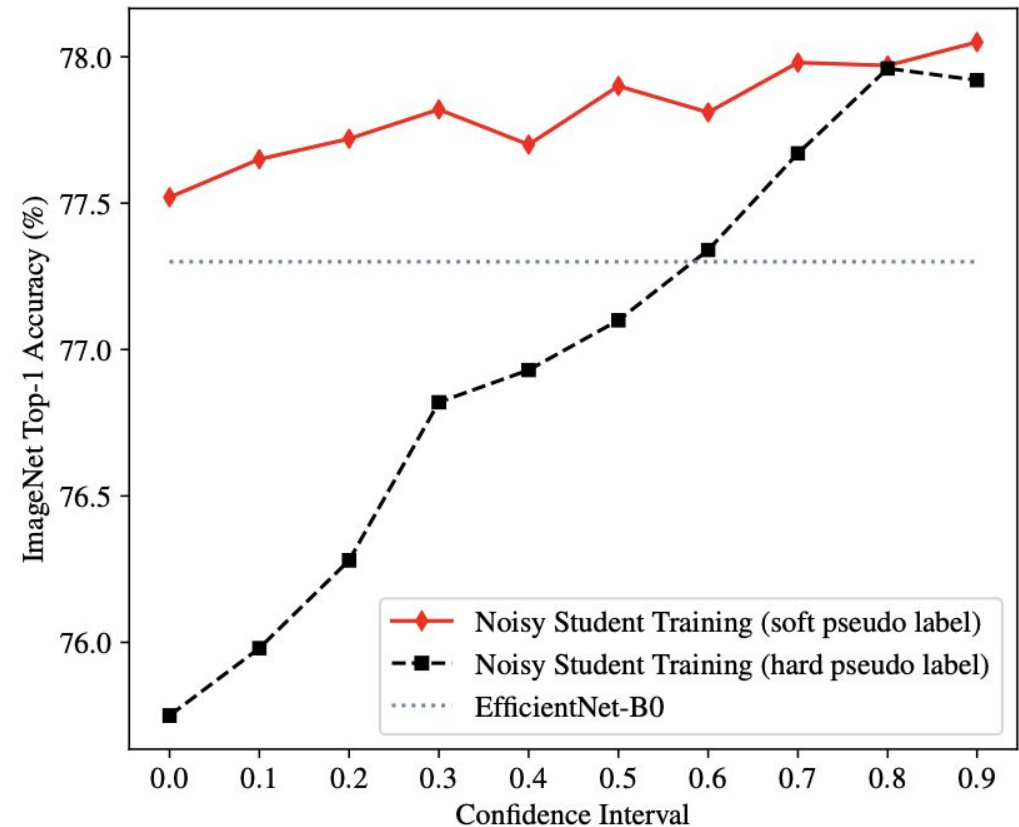


Базовая EfficientNet-B0 имеет параметры:  $\alpha=1.2$ ,  $\beta=1.1$ ,  $\gamma=1.15$ ; другие сети семейства получились масштабированием количества ресурсов.

Teacher	Teacher Acc.	Student	Student Acc.
B0	77.3%	B0	77.9%
		B1	<b>79.5%</b>
B2	80.0%	B2	80.7%
		B3	<b>82.0%</b>
B4	83.2%	B4	84.0%
		B5	<b>84.7%</b>
B7	86.9%	B7	86.9%
		L2	<b>87.2%</b>

Model	# Params	Top-1 Acc.	Top-5 Acc.
EfficientNet-B0	5.3M	77.3%	93.4%
Noisy Student Training (B0)		78.1%	94.2%
<b>Noisy Student Training (B0, L2)</b>		<b>78.8%</b>	<b>94.5%</b>
EfficientNet-B1	7.8M	79.2%	94.4%
Noisy Student Training (B1)		80.2%	95.2%
<b>Noisy Student Training (B1, L2)</b>		<b>81.5%</b>	<b>95.8%</b>
EfficientNet-B2	9.2M	80.0%	94.9%
Noisy Student Training (B2)		81.1%	95.5%
<b>Noisy Student Training (B2, L2)</b>		<b>82.4%</b>	<b>96.3%</b>
EfficientNet-B3	12M	81.7%	95.7%
Noisy Student Training (B3)		82.5%	96.4%
<b>Noisy Student Training (B3, L2)</b>		<b>84.1%</b>	<b>96.9%</b>
EfficientNet-B4	19M	83.2%	96.4%
Noisy Student Training (B4)		84.4%	97.0%
<b>Noisy Student Training (B4, L2)</b>		<b>85.3%</b>	<b>97.5%</b>
EfficientNet-B5	30M	84.0%	96.8%
Noisy Student Training (B5)		85.1%	97.3%
<b>Noisy Student Training (B5, L2)</b>		<b>86.1%</b>	<b>97.8%</b>
EfficientNet-B6	43M	84.5%	97.0%
Noisy Student Training (B6)		85.9%	97.6%
<b>Noisy Student Training (B6, L2)</b>		<b>86.4%</b>	<b>97.9%</b>
EfficientNet-B7	66M	85.0%	97.2%
Noisy Student Training (B7)		86.4%	97.9%
<b>Noisy Student Training (B7, L2)</b>		<b>86.9%</b>	<b>98.1%</b>

# Эксперименты



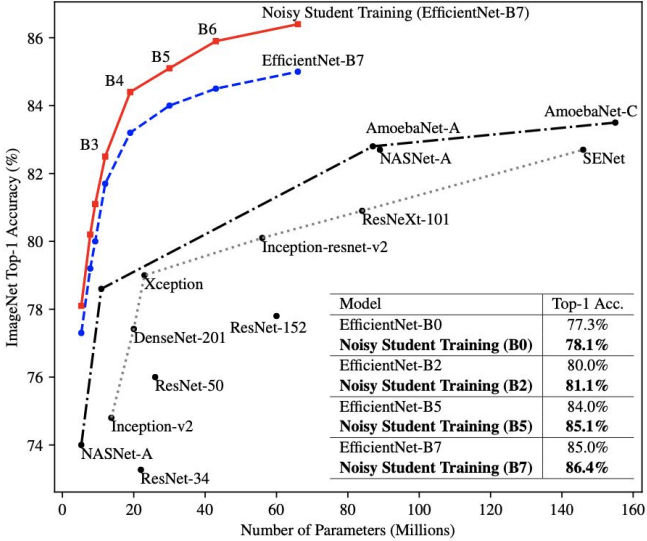
Data	1/128	1/64	1/32	1/16	1/4	1
Top-1 Acc.	83.4%	83.3%	83.7%	83.9%	83.8%	<b>84.0%</b>

Model	B0	B1	B2	B3
Supervised Learning	77.3%	79.2%	80.0%	81.7%
Noisy Student Training	<b>77.9%</b>	<b>79.9%</b>	<b>80.7%</b>	82.1%
w/o Data Balancing	77.6%	79.6%	80.6%	82.1%

# Эксперименты

Warm-start Epoch	Initializing student with teacher				No Init
	35	70	140	280	
Top-1 Acc.	77.4%	77.5%	77.7%	77.8%	<b>77.9%</b>

Method	Top-1 Acc.	Top-5 Acc.
ResNet-50	77.6%	93.8%
<b>Noisy Student Training (ResNet-50)</b>	<b>78.9%</b>	<b>94.3%</b>



Model	Dataset	Top-1 Acc.	Top-5 Acc.
EfficientNet-B0	-	77.3%	93.4%
Noisy Student Training (B0)	YFCC	79.9%	95.0%
<b>Noisy Student Training (B0)</b>	JFT	<b>78.1%</b>	<b>94.2%</b>
EfficientNet-B1	-	79.2%	94.4%
Noisy Student Training (B1)	YFCC	79.9%	95.0%
<b>Noisy Student Training (B1)</b>	JFT	<b>80.2%</b>	<b>95.2%</b>
EfficientNet-B2	-	80.0%	94.9%
Noisy Student Training (B2)	YFCC	81.0%	<b>95.6%</b>
<b>Noisy Student Training (B2)</b>	JFT	<b>81.1%</b>	95.5%
EfficientNet-B3	-	81.7%	95.7%
Noisy Student Training (B3)	YFCC	82.3%	96.2%
<b>Noisy Student Training (B3)</b>	JFT	<b>82.5%</b>	<b>96.4%</b>
EfficientNet-B4	-	83.2%	96.4%
Noisy Student Training (B4)	YFCC	84.2%	96.9%
<b>Noisy Student Training (B4)</b>	JFT	<b>84.4%</b>	<b>97.0%</b>
EfficientNet-B5	-	84.0%	96.8%
Noisy Student Training (B5)	YFCC	85.0%	97.2%
<b>Noisy Student Training (B5)</b>	JFT	<b>85.1%</b>	<b>97.3%</b>
EfficientNet-B6	-	84.5%	97.0%
Noisy Student Training (B6)	YFCC	85.4%	97.5%
<b>Noisy Student Training (B6)</b>	JFT	<b>85.6%</b>	<b>97.6%</b>
EfficientNet-B7	-	85.0%	97.2%
Noisy Student Training (B7)	YFCC	86.2%	<b>97.9%</b>
<b>Noisy Student Training (B7)</b>	JFT	<b>86.4%</b>	<b>97.9%</b>

# Условия экспериментов



Размеченный датасет - ImageNet 2012 ILSVRC.

Неразмеченный датасет - JFT.

Для каждого класса имеется 130.000 немаркированных изображений.

Batch size = 2048 (но различий между 512, 1024 и 2048 не наблюдалось).

Обучение длится 350 эпох для меньших моделей, 700 - для больших.

Learning Rate устанавливается равным 0,128, а затем умножается на 0,97.

Для EfficientNet-L2 первые 350 эпох обучение на меньшем разрешении (224x224), затем модель переучивается под большее (299x299).

Stochastic depth :  $p=0,2$  для последнего слоя. Линейно возрастает к первому слою.

Dropout :  $p = 0,5$  для последнего слоя.

Параметры аугментации:  $N = 2$ ;  $M = 27$ .



# Эксперименты

Лучшая модель:

- 3 итерации алгоритма
- Учитель - EfficientNet-B7, остальные - EfficientNet-L2

Method	# Params	Extra Data	Top-1 Acc.	Top-5 Acc.
ResNet-50 [30]	26M	-	76.0%	93.0%
ResNet-152 [30]	60M	-	77.8%	93.8%
DenseNet-264 [36]	34M	-	77.9%	93.9%
Inception-v3 [81]	24M	-	78.8%	94.4%
Xception [15]	23M	-	79.0%	94.5%
Inception-v4 [79]	48M	-	80.0%	95.0%
Inception-resnet-v2 [79]	56M	-	80.1%	95.1%
ResNeXt-101 [92]	84M	-	80.9%	95.6%
PolyNet [100]	92M	-	81.3%	95.8%
SENet [35]	146M	-	82.7%	96.2%
NASNet-A [104]	89M	-	82.7%	96.2%
AmoebaNet-A [65]	87M	-	82.8%	96.1%
PNASNet [50]	86M	-	82.9%	96.2%
AmoebaNet-C [17]	155M	-	83.5%	96.5%
GPipe [38]	557M	-	84.3%	97.0%
EfficientNet-B7 [83]	66M	-	85.0%	97.2%
EfficientNet-L2 [83]	480M	-	85.5%	97.5%
ResNet-50 Billion-scale [93]	26M	3.5B images labeled with tags	81.2%	96.0%
ResNeXt-101 Billion-scale [93]	193M		84.8%	-
ResNeXt-101 WSL [55]	829M		85.4%	97.6%
FixRes ResNeXt-101 WSL [86]	829M		86.4%	98.0%
Big Transfer (BiT-L) [43] <sup>†</sup>	928M	300M weakly labeled images from JFT	87.5%	98.5%
<b>Noisy Student Training (EfficientNet-L2)</b>	480M	300M unlabeled images from JFT	<b>88.4%</b>	<b>98.7%</b>

Method	Top-1 Acc.	Top-5 Acc.
ResNet-101 [32]	4.7%	-
ResNeXt-101 [32] (32x4d)	5.9%	-
ResNet-152 [32]	6.1%	-
ResNeXt-101 [32] (64x4d)	7.3%	-
DPN-98 [32]	9.4%	-
ResNeXt-101+SE [32] (32x4d)	14.2%	-
ResNeXt-101 WSL [55, 59]	61.0%	-
EfficientNet-L2	49.6%	78.6%
<b>Noisy Student Training (L2)</b>	<b>83.7%</b>	<b>95.2%</b>

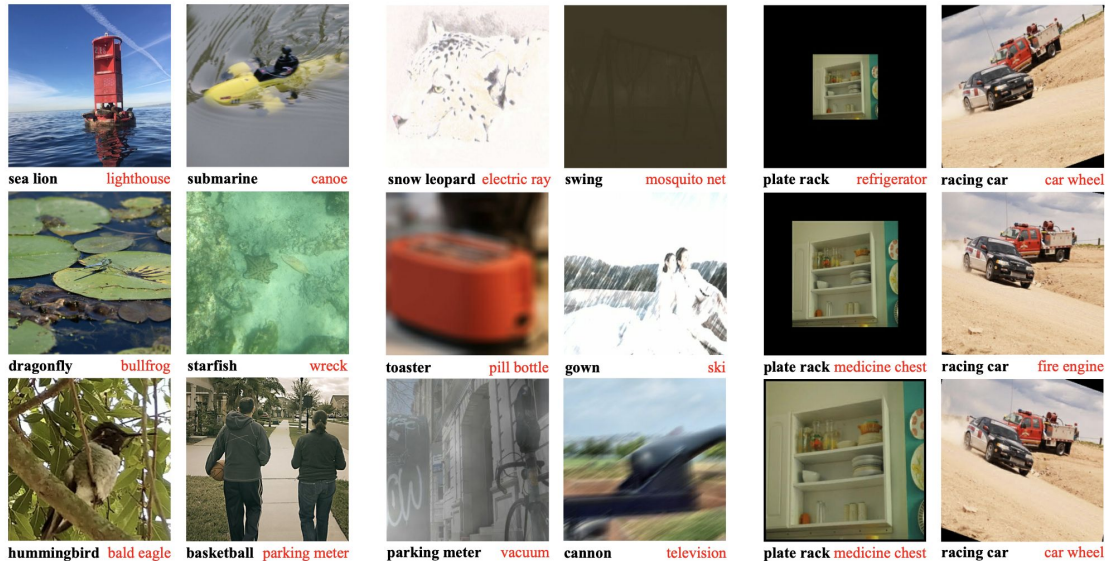
Table 3: Robustness results on ImageNet-A.

Method	Res.	Top-1 Acc.	mCE
ResNet-50 [31]	224	39.0%	76.7
SIN [23]	224	45.2%	69.3
Patch Gaussian [51]	299	52.3%	60.4
ResNeXt-101 WSL [55, 59]	224	-	45.7
EfficientNet-L2	224	62.6%	47.5
Noisy Student Training (L2)	224	76.5%	30.0
EfficientNet-L2	299	66.6%	42.5
<b>Noisy Student Training (L2)</b>	299	<b>77.8%</b>	<b>28.3</b>

Table 4: Robustness results on ImageNet-C. mCE is the weighted average of error rate on different corruptions, with AlexNet’s error rate as a baseline (lower is better).

Method	Res.	Top-1 Acc.	mFR
ResNet-50 [31]	224	-	58.0
Low Pass Filter Pooling [99]	224	-	51.2
ResNeXt-101 WSL [55, 59]	224	-	27.8
EfficientNet-L2	224	80.4%	27.2
Noisy Student Training (L2)	224	85.2%	14.2
EfficientNet-L2	299	81.6%	23.7
<b>Noisy Student Training (L2)</b>	299	<b>86.4%</b>	<b>12.2</b>

Table 5: Robustness results on ImageNet-P, where images are generated with a sequence of perturbations. mFR measures the model’s probability of flipping predictions under perturbations with AlexNet as a baseline (lower is better).



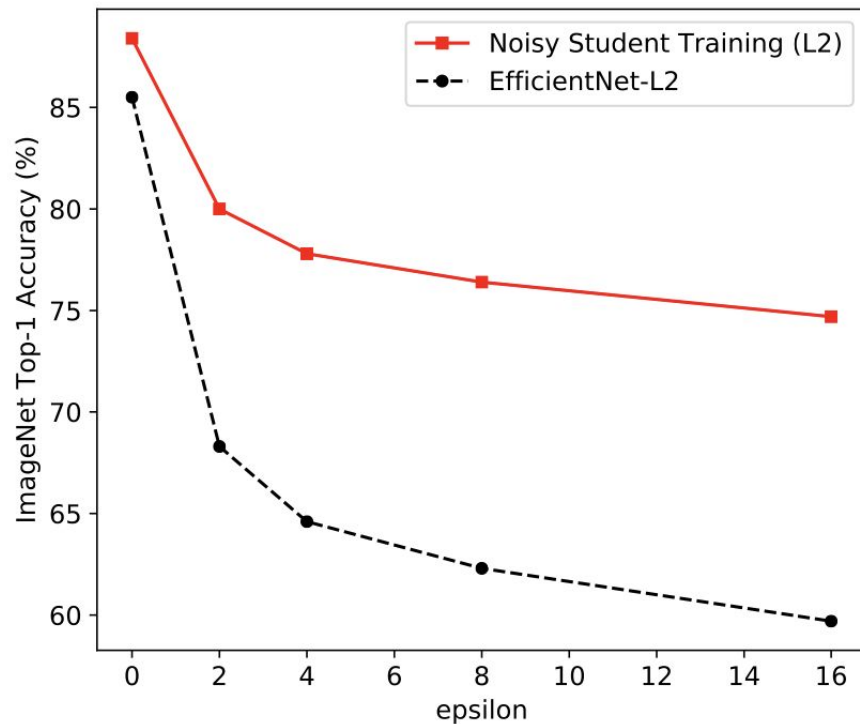
(a) ImageNet-A

(b) ImageNet-C

(c) ImageNet-P

**FGSM-атаки** : в данные добавляется шум, рассчитываемый из эpsilon, умноженного на знак градиента функции потерь входа

Для PGD-атак улучшение составило с 1,1% до 4,4%





---

**Спасибо за внимание!**

<https://arxiv.org/pdf/1911.04252.pdf> - статья Noisy Student

<https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf> -

<https://www.youtube.com/watch?v=q7PirmGNx5A> - обзор статьи Noisy Student

<https://towardsdatascience.com/review-stochastic-depth-image-classification-a4e225807f4a> - stochastic depth

<https://arxiv.org/pdf/1909.13719.pdf> - статья RandAugmentation

<https://towardsdatascience.com/simple-image-data-augmentation-technics-to-mitigate-overfitting-in-computer-vision-2a6966f51af4> - статья про аугментации

<https://www.youtube.com/watch?v=Zzt9i3gDueE> - урок по RandAugment

<https://proglib.io/p/issleduem-arhitektury-svertochnyh-neyronnyh-setey-s-po-moshchyu-fast-ai-2020-12-28> - разбор статьи efficient Net

<https://arxiv.org/pdf/1905.11946.pdf> - статья про EfficientNet

<https://www.youtube.com/watch?v=zbp9egDwwRc> - видеоурок про efficientNet