Saliency Mix :
A Saliency guided data augmentation strategy for better regularization

Group 10 : MyeongJae Cho, Yujin Kim, Ye-eun Shin

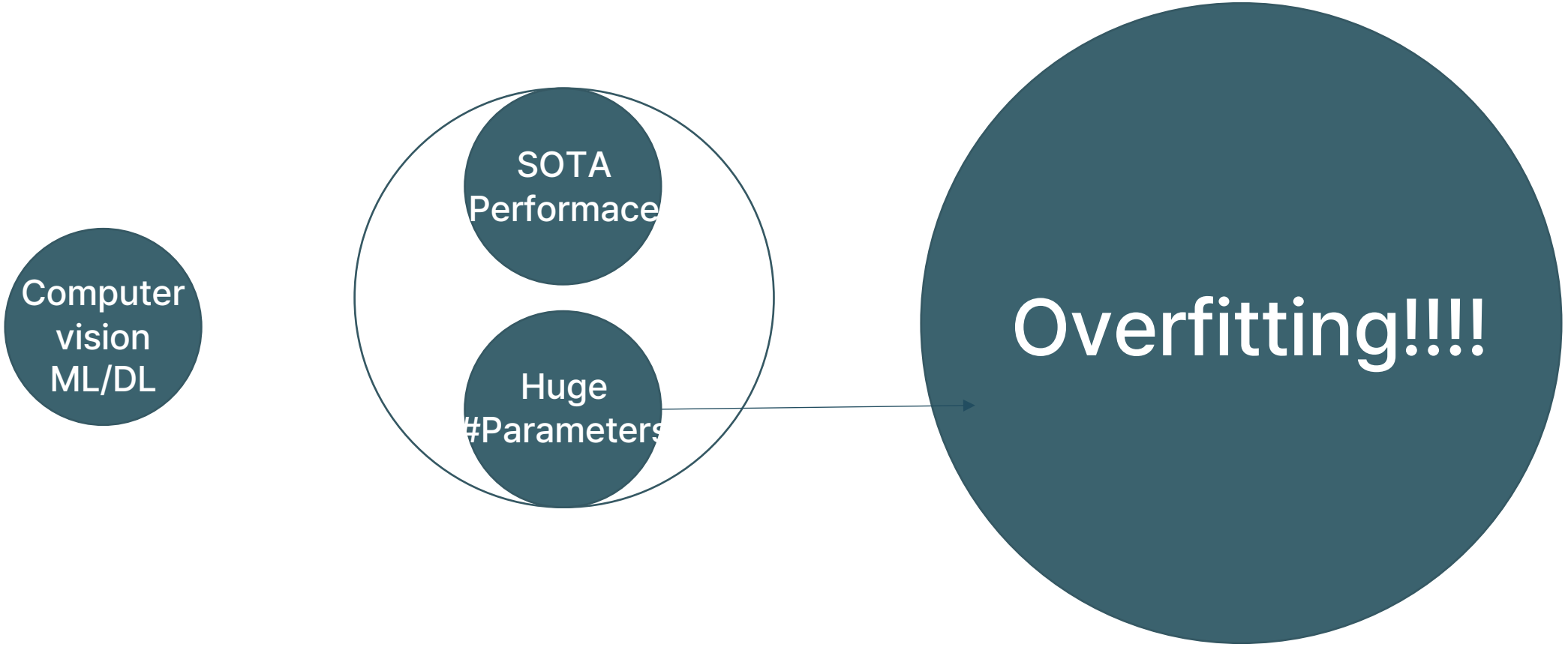
Contents

1

Introduction & Background

Part 1

Introduction



Introduction

Overfitting

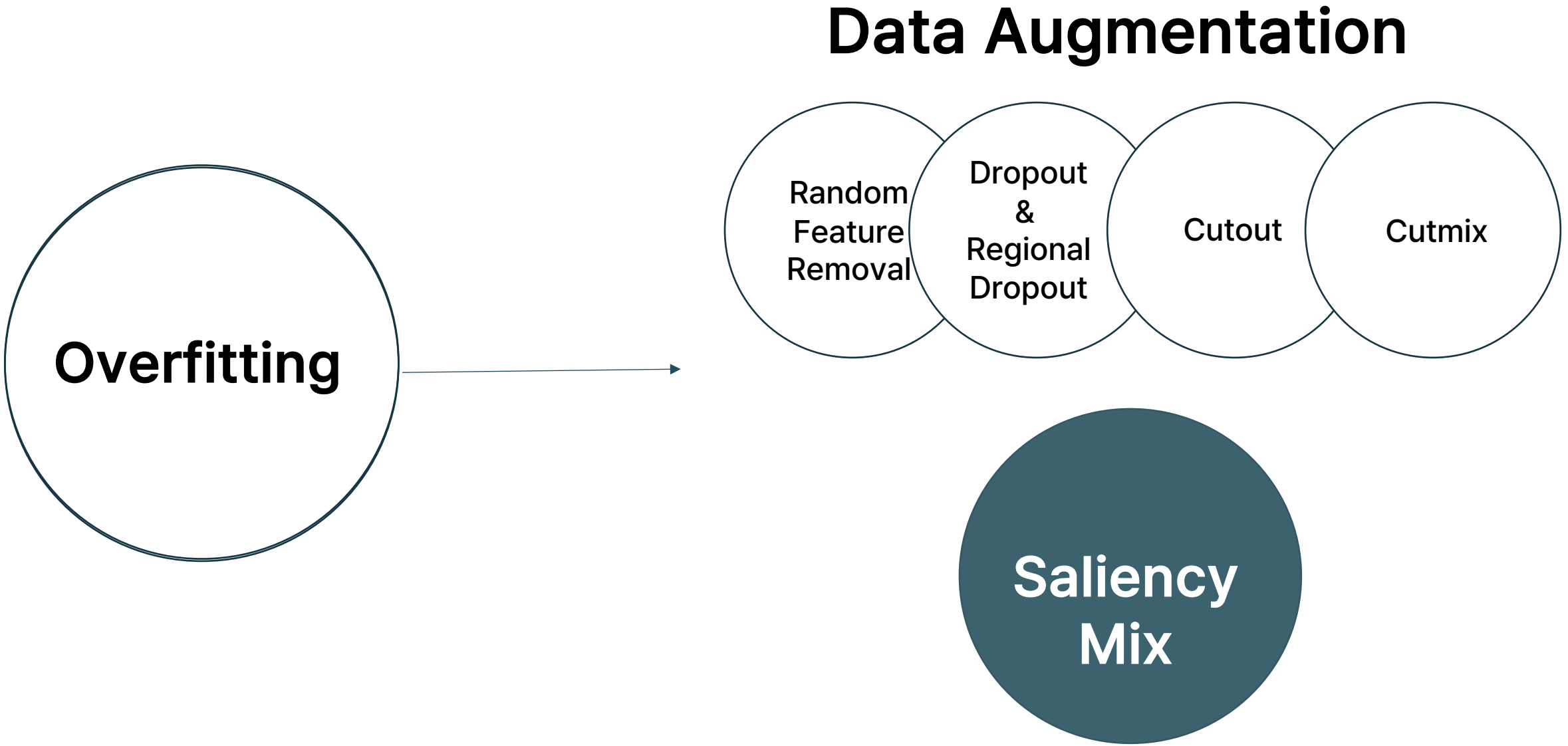
Data Augmentation

- Random Feature Removal
- Dropout & Regional Dropout
- Cutout
- Cutmix

👉 Don’t care about informative pixel

Target Image	Source Image	Augmented Image	
			
Mixed label for randomly mixed images		Dog - 80% & Cat 20% ?	Dog - 80% & Cat 20% ?

Introduction



2

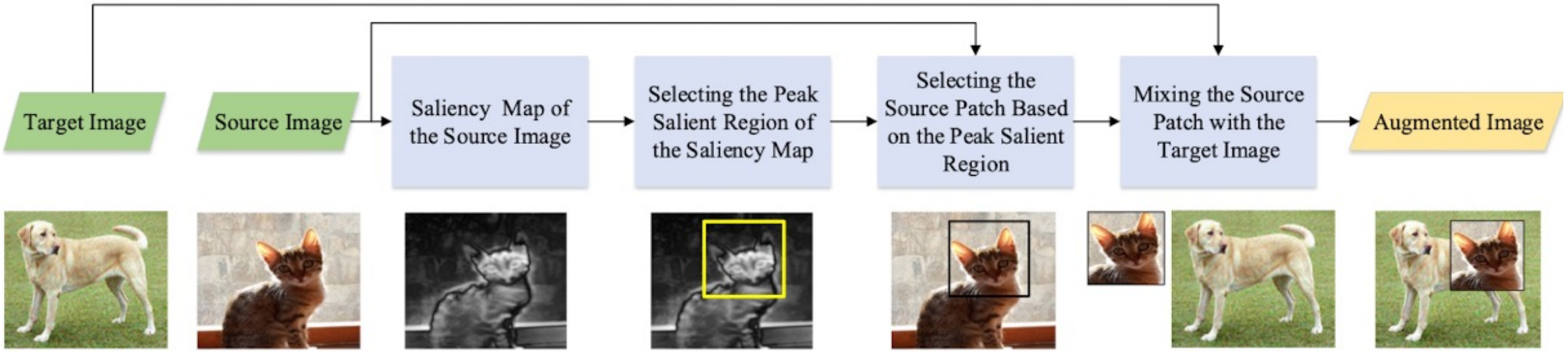
Methods

Data augmentation

: techniques to artificially increase the amount of data
by generating new data points from existing data

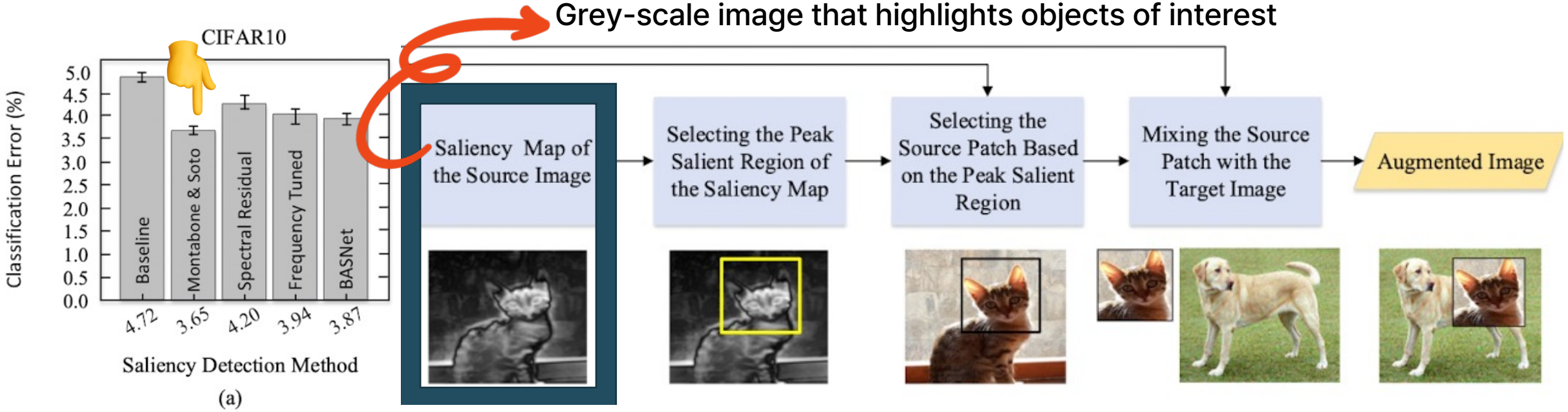
Method

Process of **Saliency Mix** data augmentation



Method

Process of **Saliency Mix** data augmentation



👉 $I_{vs} = f(I_s)$

I_s : *source image*

f : *salinecy model*

I_{vs} : *saliency map*

👉 $i, j = argmax(I_{vs})$

👉 $I_{vs}^{i,j} - th$ = pixel that has **maximum intensity value**

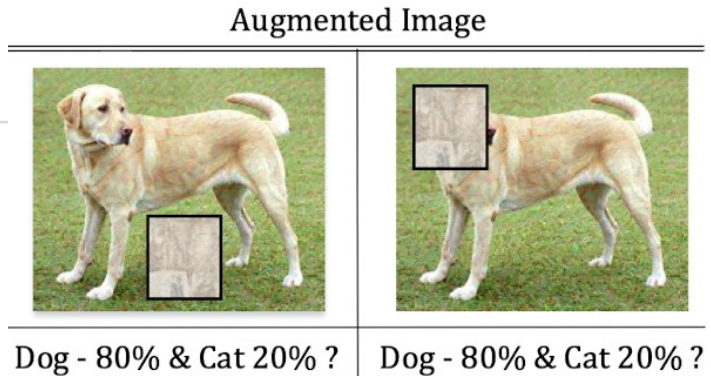
Method

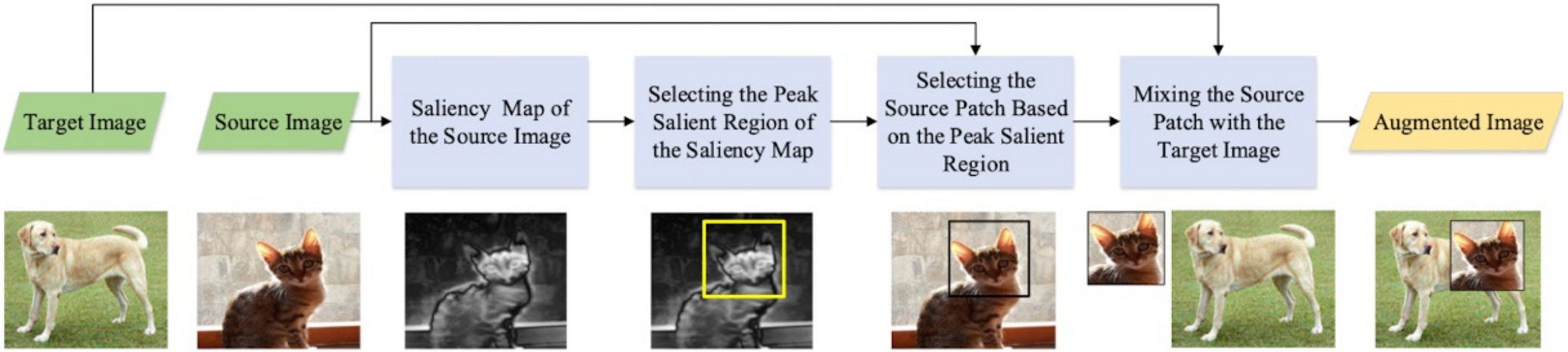
Process of **Saliency Mix** data augmentation

Based on $I_{vs}^{i,j} - th$

Method

Process of **Saliency Mix** data augmentation

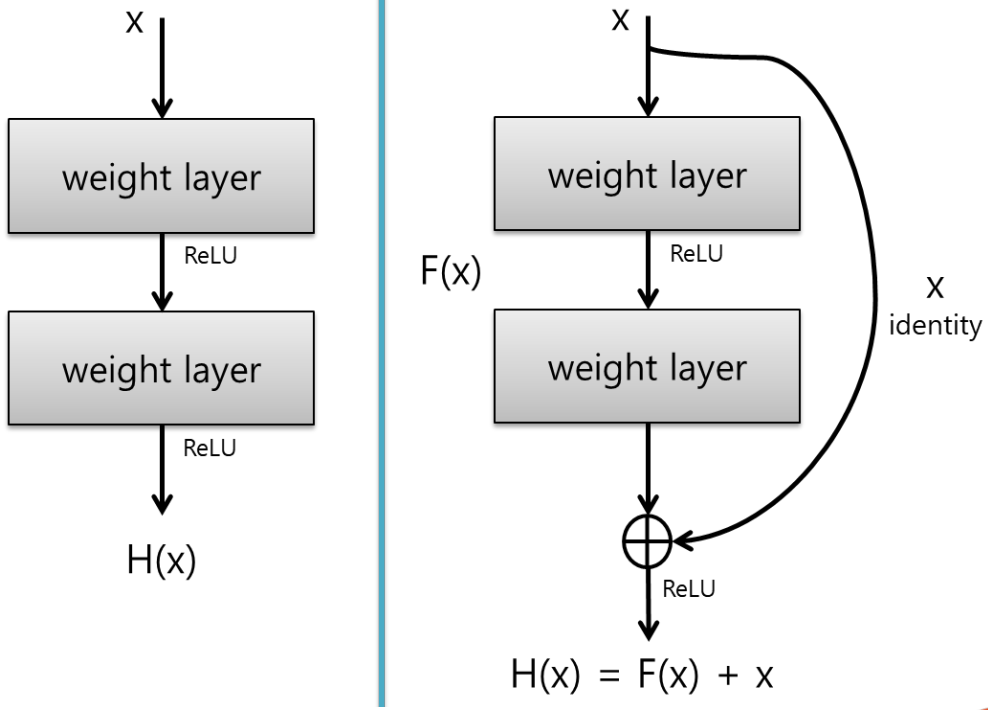




👉 Proper feature representation!

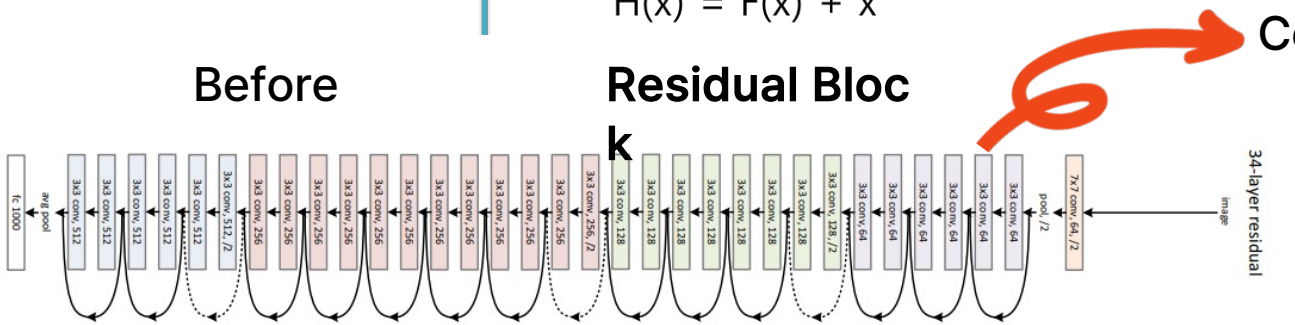
Method

What is ResNet?(Residual neural Network)



Before

Residual Block



👉 deeper layer
& solve vanishing gradient proble

3

Evaluation

Part 3

Paper's training methods

CIFAR-10 & CIFAR-100

"+" sign means that the traditional data augmentation techniques are also used

Method	Top-1 Error (%)			
	CIFAR-10	CIFAR-10+	CIFAR-100	CIFAR-100+
ResNet-18 (Baseline)	10.63± 0.26	4.72± 0.21	36.68± 0.57	22.46± 0.31
ResNet-18 + Cutout	9.31±0.18	3.99±0.13	34.98±0.29	21.96±0.24
ResNet-18 + CutMix	9.44±0.34	3.78±0.12	34.42±0.27	19.42±0.23
ResNet-18 + SaliencyMix	**7.59±0.22**	**3.65±0.10**	**28.73±0.13**	**19.29±0.21**
ResNet-50 (Baseline)	12.14±0.95	4.98±0.14	36.48±0.50	21.58±0.43
ResNet-50 + Cutout	8.84±0.77	3.86±0.25	32.97±0.74	21.38±0.69
ResNet-50 + CutMix	9.16±0.38	3.61±0.13	31.65±0.61	18.72±0.23
ResNet-50 + SaliencyMix	**6.81±0.30**	**3.46±0.08**	**24.89±0.39**	**18.57±0.29**
WideResNet-28-10 (Baseline)	6.97±0.22	3.87±0.08	26.06±0.22	18.80±0.08
WideResNet-28-10 + Cutout	5.54±0.08	3.08±0.16	23.94±0.15	18.41±0.27
WideResNet-28-10 + AutoAugment	-	**2.60±0.10**	-	17.10±0.30
WideResNet-28-10 + PuzzleMix (200 epochs)	-	-	-	**16.23**
WideResNet-28-10 + CutMix	5.18±0.20	2.87±0.16	23.21±0.20	16.66±0.20
WideResNet-28-10 + SaliencyMix	**4.04±0.13**	2.76±0.07	**19.45±0.32**	16.56±0.17

ImageNet

Method	Top-1 Error (%)	Top-5 Error (%)
ResNet-50 (Baseline)	23.68	7.05
ResNet-50 + Cutout	22.93	6.66
ResNet-50 + StochasticDepth	22.46	6.27
ResNet-50 + Mixup	22.58	6.40
ResNet-50 + Manifold Mixup	22.50	6.21
ResNet-50 + AutoAugment	22.40	6.20
ResNet-50 + DropBlock	21.87	5.98
ResNet-50 + CutMix	21.40	5.92
ResNet-50 + PuzzleMix	**21.24**	**5.71**
ResNet-50 + SaliencyMix	**21.26**	**5.76**
ResNet-101 (Baseline)	21.87	6.29
ResNet-101 + Cutout	20.72	5.51
ResNet-101 + Mixup	20.52	5.28
ResNet-101 + Cutmix	20.17	5.24
ResNet-101 + SaliencyMix	**20.09**	**5.15**

Highlights means that we selected as experimental methods because codes were available

Part 3

Our Experimental Setup

The training method we chose:

- ResNET 18
- ResNET 50
- WideResNet-28-10

CIFAR10 Dataset

- CutOut + Traditional Data Augmentation
- SaliecyMix + Traditional Data Augmentation
- Baseline (Only traditional augmentations of flipping and cropping)

3 models x 1 dataset x 3 version of augmentation = **total 9 models**

Our Experimental Setup

EXPERIMENTAL SETUP
200 epochs
128 batch size
SGD
Nesterov momentum 0.9
weight decay 5e-4
Initial learning rate 0.1, reduced by 0.2x at 60, 120, 160 epoch
Normalization using mean, std of each image channel

Total 9 model for training

👉 Evaluate Image **classification** performance

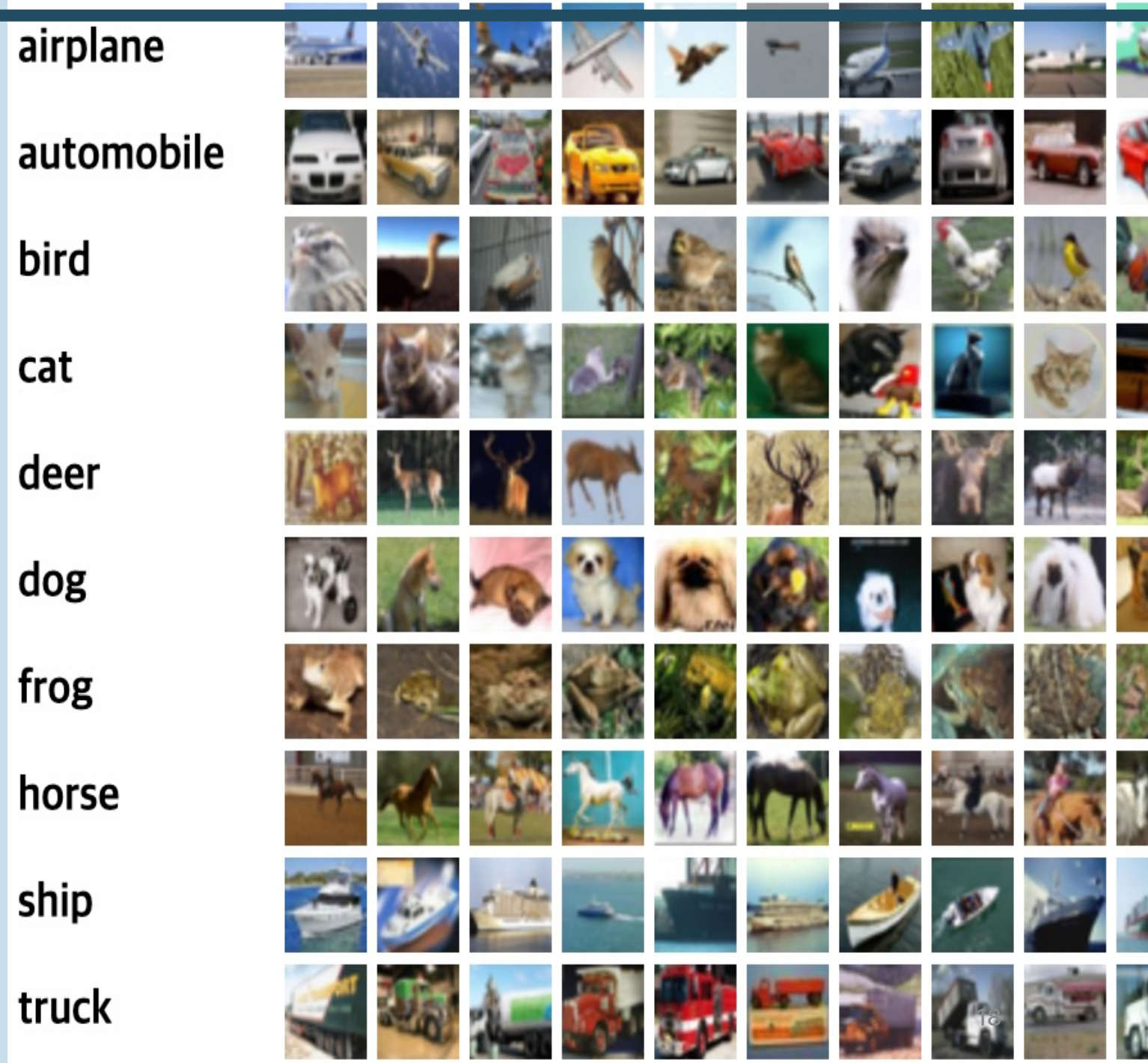
&

Computational Complexity

Part 3

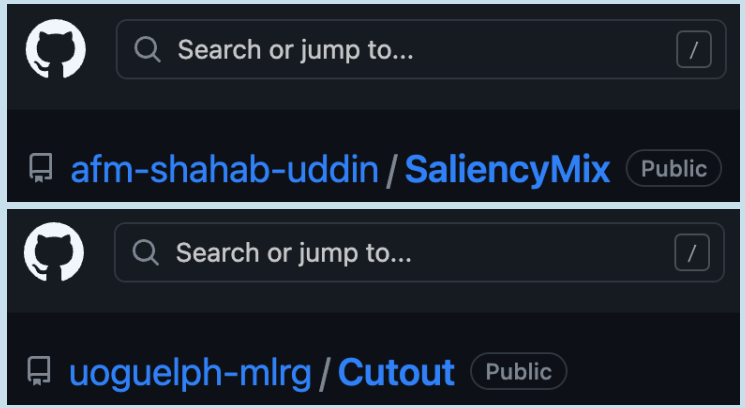
CIFAR10

- 60000 32 x 32 color images
- 10 labels
- 5000 training images, 1000 test images for each class



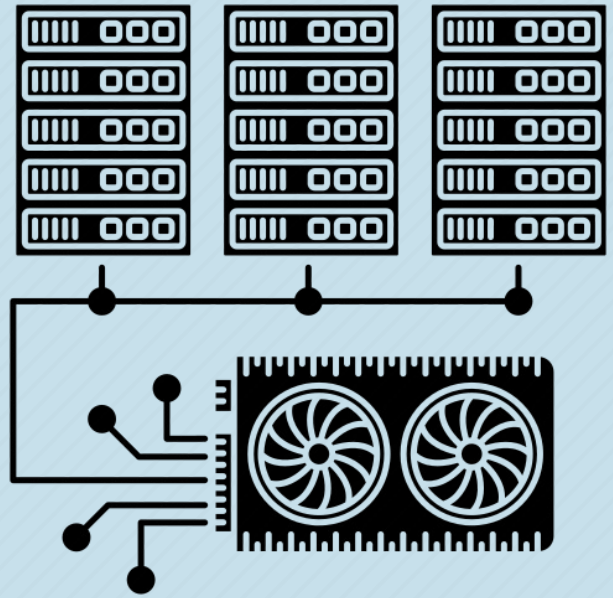
Part 3

Training model



1. Git clone repository for Saliency Mix and CutOut
2. Set environment in GPU server using anaconda3
3. Download CIFAR10 Dataset
4. Code debugging & modification to evaluate model and training time

5. Train models in in GPU server
6. Efficient training with GPU allocation (CUDA option) and background tasks (nohup)
7. Get test report & checkpoints & Training time



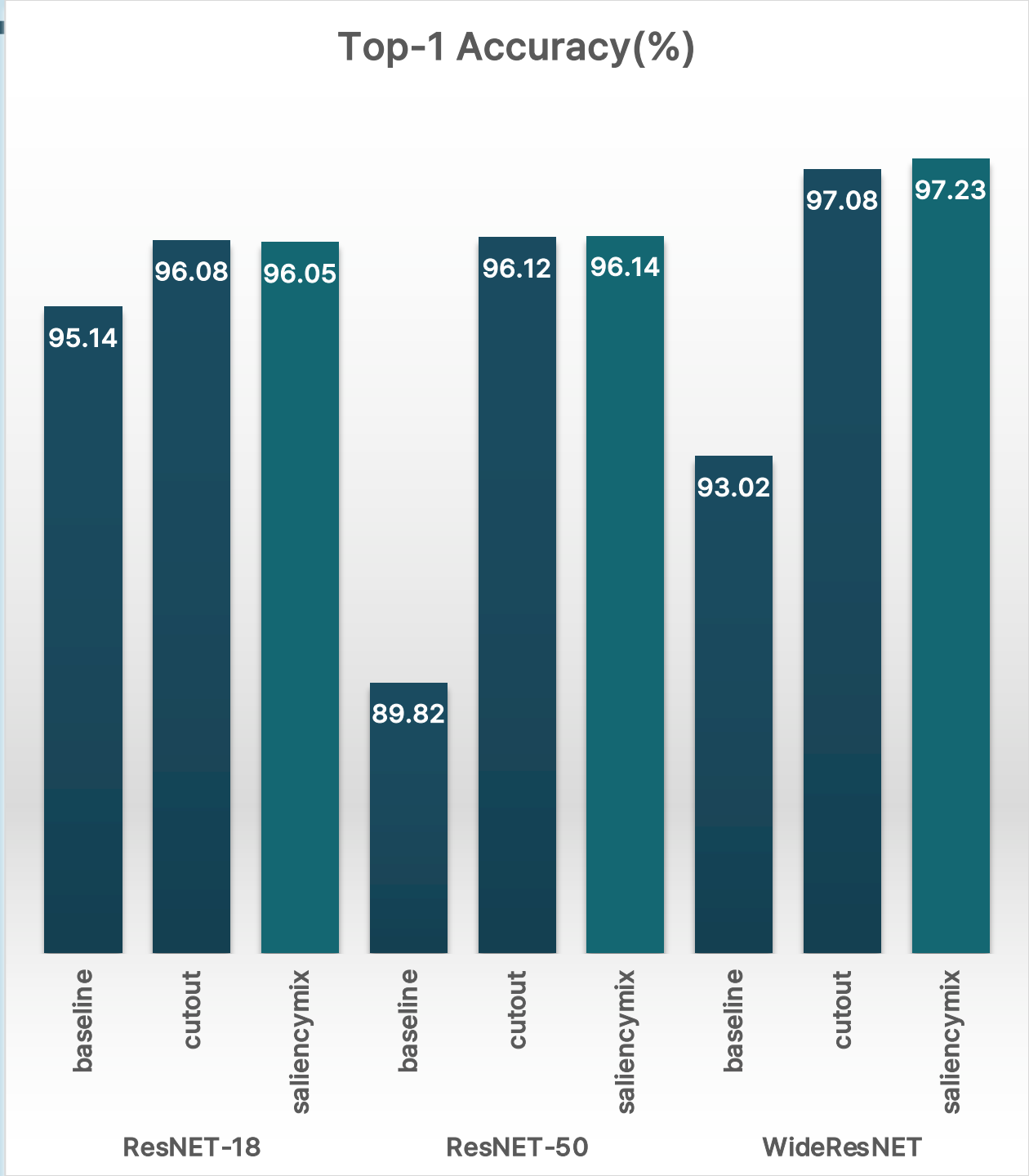
- cifar10_res18_baseline_2.out
- cifar10_res18_cutout_2.out
- cifar10_resnet18_baseline_2.csv
- cifar10_resnet18_baseline.csv
- cifar10_resnet18_baseline.out
- cifar10_resnet18_baseline.pt
- cifar10_resnet18_cutout_2.csv
- cifar10_resnet18_cutout_2.pt
- cifar10_resnet18_cutout.csv
- cifar10_resnet18_cutout.pt
- cifar10_resnet18_saliencymix_2.csv
- cifar10_resnet18_saliencymix_best_accuracy_2.txt
- cifar10_resnet18_saliencymix.csv
- cifar10_resnet18_saliencymix.pt
- cifar10_resnet50_baseline.csv
- cifar10_resnet50_baseline.log
- cifar10_resnet50_baseline.pt
- cifar10_resnet50_cutout.csv
- cifar10_resnet50_cutout.log
- cifar10_resnet50_cutout.pt
- cifar10_resnet50_saliencymix_best_acc.txt
- cifar10_resnet50_saliencymix.csv
- cifar10_resnet50_saliencymix.pt
- cifar10_wideresnet_baseline.csv
- cifar10_wideresnet_baseline.log
- cifar10_wideresnet_baseline.pt
- cifar10_wideresnet_cutout.csv
- cifar10_wideresnet_cutout.log
- cifar10_wideresnet_cutout.pt
- cifar10_wideresnet_saliencymix_best_accuracy.txt
- cifar10_wideresnet_saliencymix.csv
- cifar10_wideresnet_saliencymix.pt
- running_time_cifar10_resnet18_baseline.txt
- running_time_cifar10_resnet18_cutout.txt
- running_time_saliencymixcifar10_resnet18.txt
- running_time_saliencymixcifar10_wideresnet.txt

Part 3

Our Results

Model	Augmentation	Top-1 Accuracy(%)	Top-1 Error (%)
ResNET-18	Baseline	95.14	4.86
	CutOut	96.08	3.92
	SaliencyMix	96.05	3.95
ResNET-50	Baseline	95.46	4.54
	CutOut	96.12	3.88
	SaliencyMix	96.14	3.86
WideResNET	Baseline	96.28	3.72
	CutOut	97.08	2.92
	SaliencyMix	97.23	2.77

Model	Augmentation	Training time (min)	Training time (hour)
ResNET18	Baseline	44.66	0.74
	CutOut	59.28	0.99
	SaliencyMix	47.00	0.78



4

Code Demo

5

Challeneges

Part 5

Challenges

Many model training for performance comparison.
Very unstable in Colab environment

High Capacity of ImageNet Dataset needs extremely high computational cost

✓ Set environment in GPU server
✓ Background training with **nohup** command
✓ efficient usage of GPU source with `CUDA_VISIBLE_DEVICE` command

✓ Use only CIFAR10 dataset to compare the performance impact of each augmentation approach

Part 5

Discussion

Model	Augmentation	Top-1 Accuracy(%)	Top-1 Error (%)
ResNET-18	Baseline	95.14	4.86
	CutOut	96.08	3.92
	SaliencyMix	96.05	3.95
ResNET-50	Baseline	95.46	4.54
	CutOut	96.12	3.88
	SaliencyMix	96.14	3.86
WideResNET	Baseline	96.28	3.72
	CutOut	97.08	2.92
	SaliencyMix	97.23	2.77

Model	Augmentation	Training time (min)	Training time (hour)
ResNET18	Baseline	44.66	0.74
	CutOut	59.28	0.99
	SaliencyMix	47.00	0.78

No dramatic performance difference

- Augmentation methods not always guarantee better performance
- The basic process randomly selecting and mixing images is the same
- Performance variation present

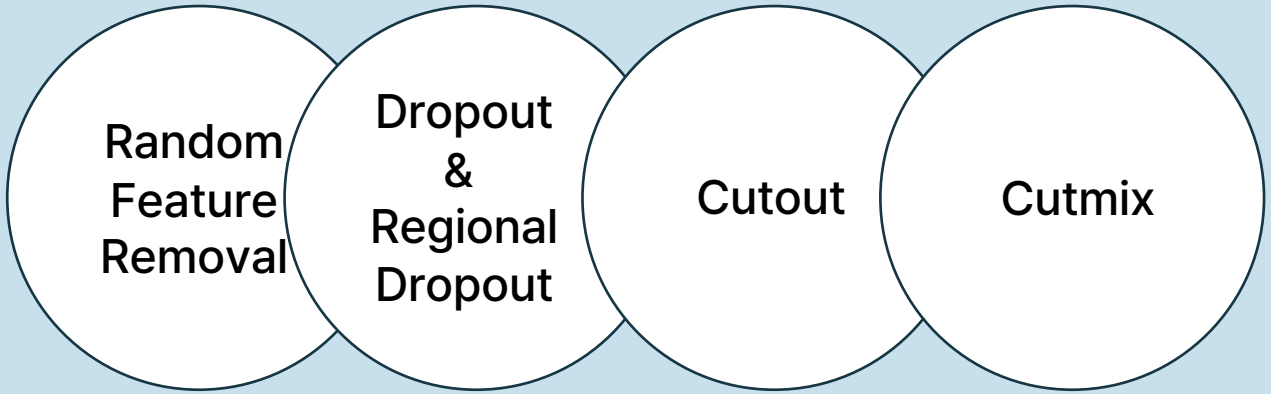
✓ **Faster than CutOut**

✓ **Meaningful to reduce the variation of performance (loss of information) that can occur in CutOut.**

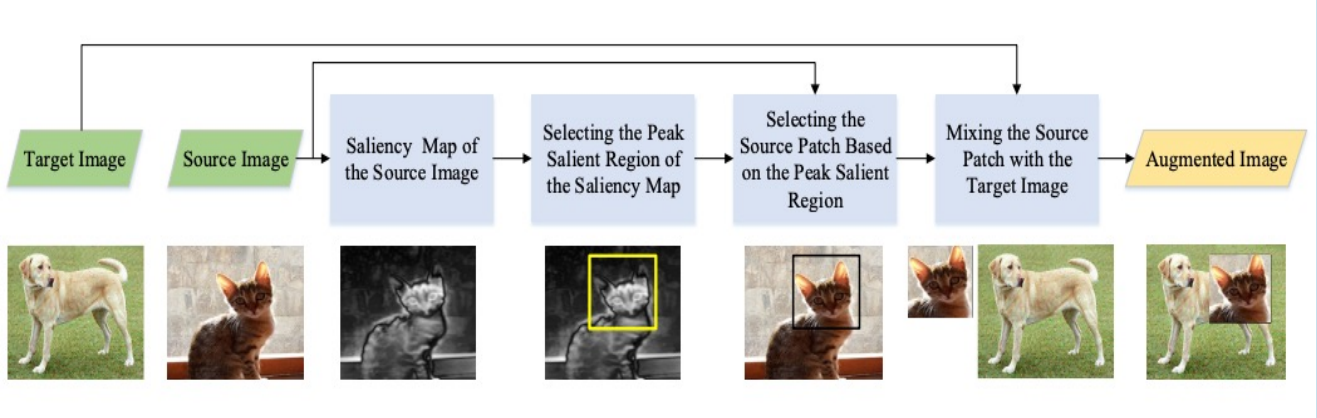
6

Conclusion

Summary



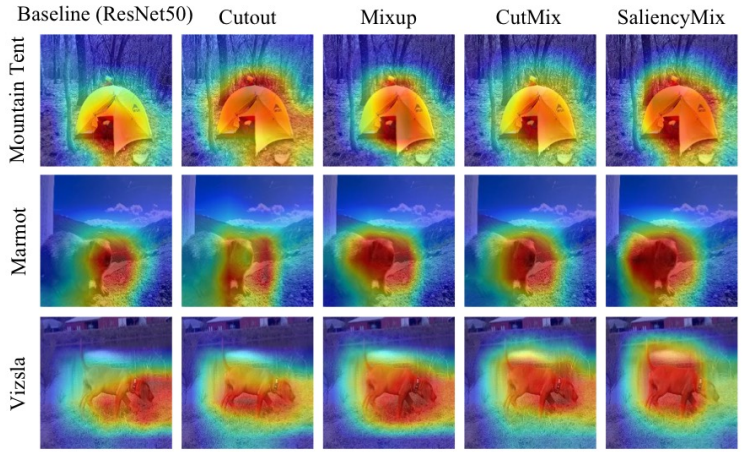
👉 Don’t care about informative pixel



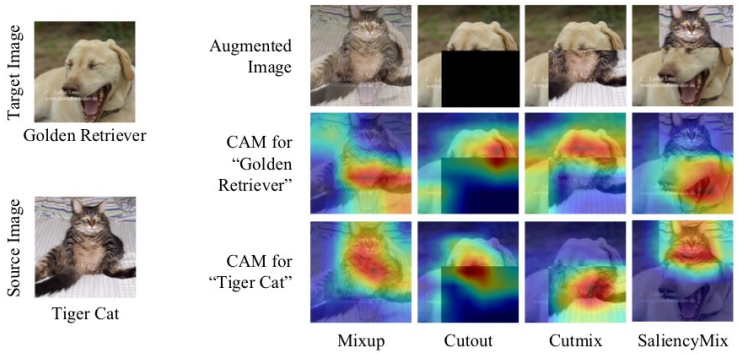
1. Extract the saliency map of the source image that highlights important objects
2. Select the patch surrounding the peak saliency region of the source image
3. Mix with target image, mix labels

✓ Unlike Regional Dropout, CutOut, CutMix, prevent informative pixel loss because patches are not selected randomly

Limitations



(a) CAM visualizations on un-augmented images. The proposed data augmentation method guides the model to precisely focus on target object.



(b) CAM visualizations on augmented images. Left most column shows the original images, top row shows the input augmented images by different methods, middle and bottom rows show the CAM for 'Golden Retriever', and 'Tiger Cat' class, respectively. The proposed method and Yun et al. (2019) improves the localization ability of the model.

Figure 4: Class activation map (CAM) for models that are trained with various data augmentation techniques. The images are randomly taken from ImageNet (Olga et al., 2015) validation set.

Other benchmarks in paper : Object Detection, CAM analysis, Adversarial Attack..
Not implemented in GitHub.
It was very difficult for us to implement them

Appendix

Paper Setup Vs. Project setup

Paper Experimental Setup	Project Experimental Setup
200 epochs	200 epochs
128 batch size	128 batch size
SGD	SGD
Nesterov momentum 0.9	Nesterov momentum 0.9
weight decay 5e-4	weight decay 5e-4
Initial learning rate 0.1, reduced by 0.2x at 60, 120, 160 epoch	Initial learning rate 0.1, reduced by 0.2x at 60, 120, 160 epoch
Normalization using mean, std of each image channel	Normalization using mean, std of each image channel

Results vs. Paper Results

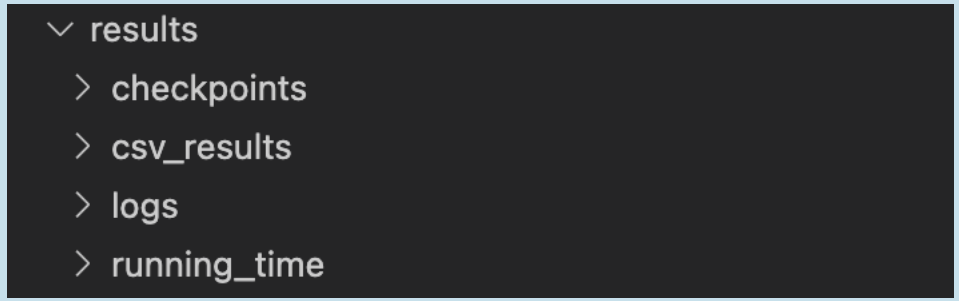
We succeeded in reproducing the results of the paper.

Model	Augmentation	Top-1 Accuracy(%)	Top-1 Error (%)	Top-1 Error in paper
ResNET-18	Baseline	95.14	4.86	4.72± 0.21
	CutOut	96.08	3.92	3.99±0.13
	SaliencyMix	96.05	3.95	3.65±0.10
ResNET-50	Baseline	95.46	4.54	4.98±0.14
	CutOut	96.12	3.88	3.86±0.25
	SaliencyMix	96.14	3.86	3.46±0.08
WideResNET	Baseline	96.28	3.72	3.87±0.08
	CutOut	97.08	2.92	3.08±0.16
	SaliencyMix	97.23	2.77	2.76±0.07

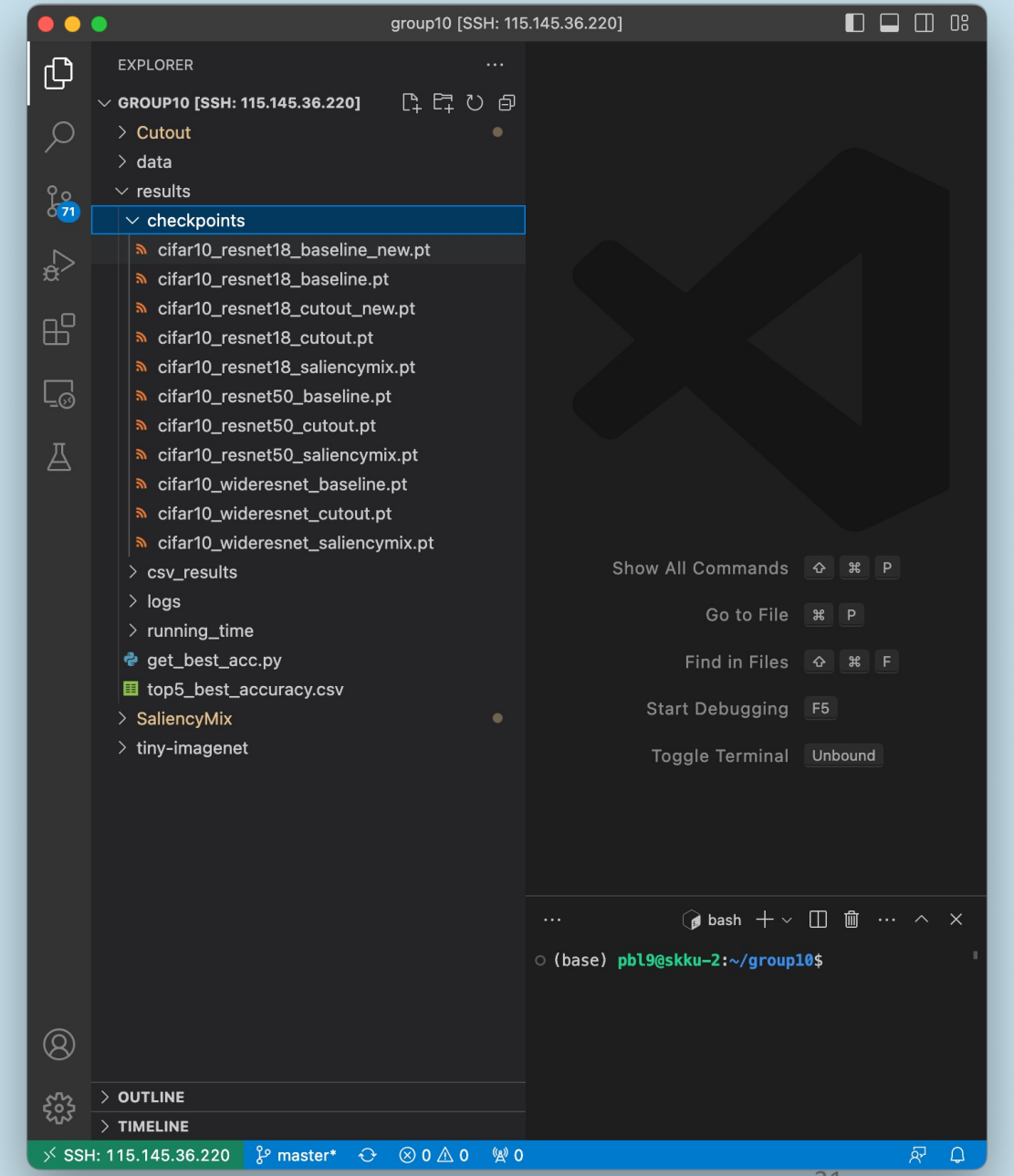
Model	Augmentation	Training time (min)	Training time (hour)	Training time in paper
ResNET18	Baseline	44.66	0.74	0.83
	CutOut	59.28	0.99	0.84
	SaliencyMix	47.00	0.78	0.91

How to see the results? (1)

Our results are divided into 4 directories in ‘results’ directory.

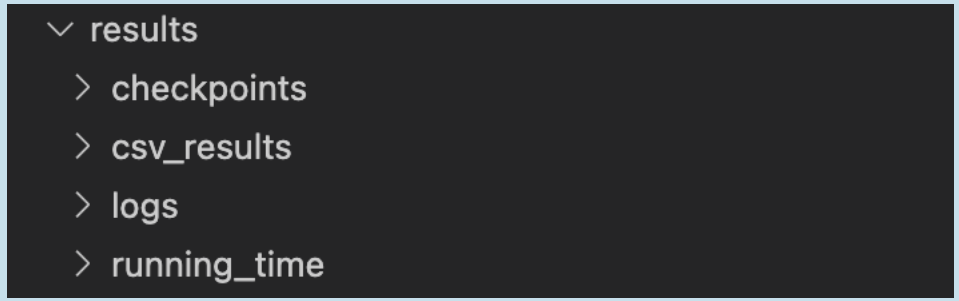


1. checkpoints : ‘.pt’ files saving the result of weights of training each model
2. csv_results
3. logs
4. running_time

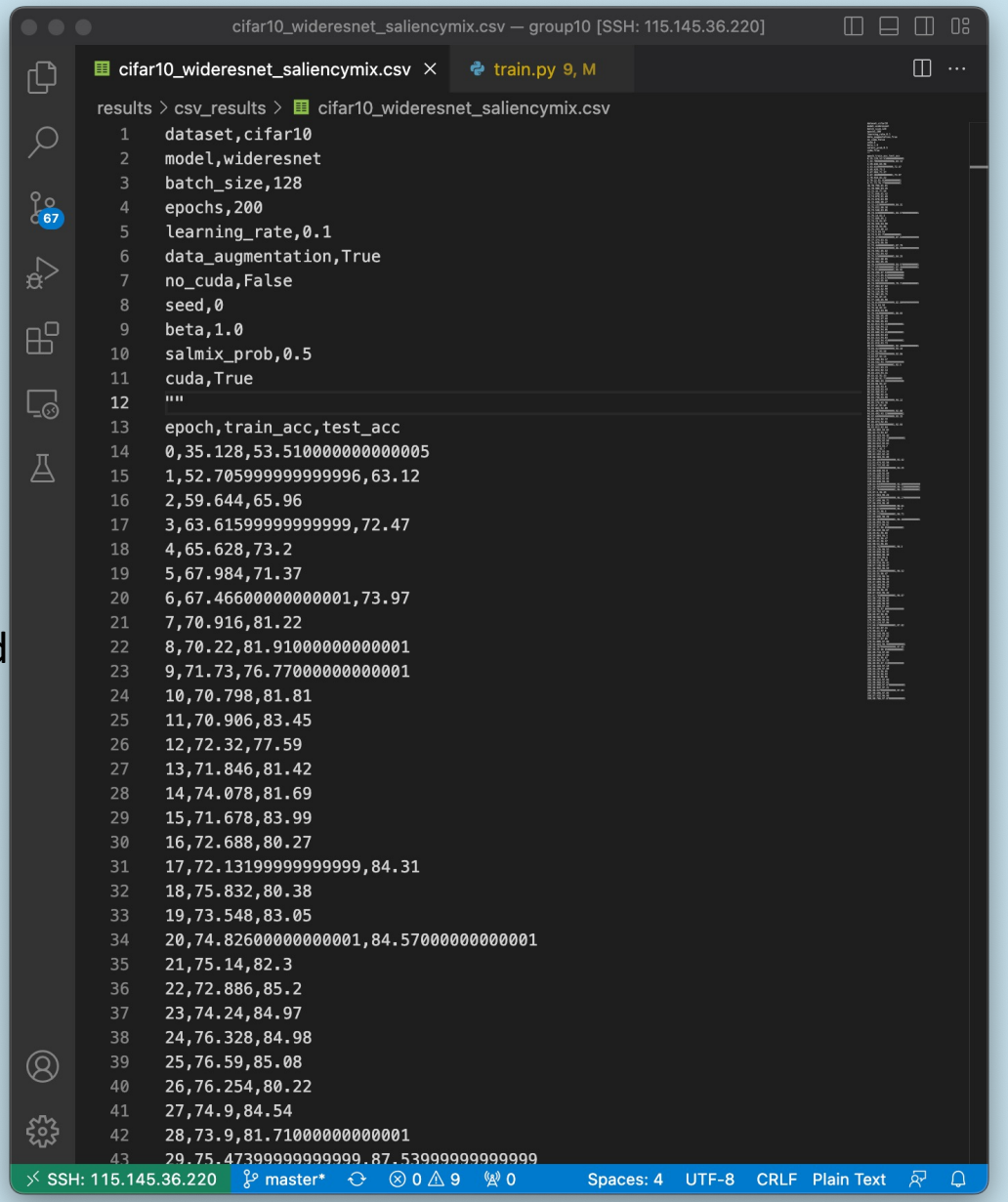


How to see the results? (2)

Our results are divided into 4 directories in ‘results’ directory.

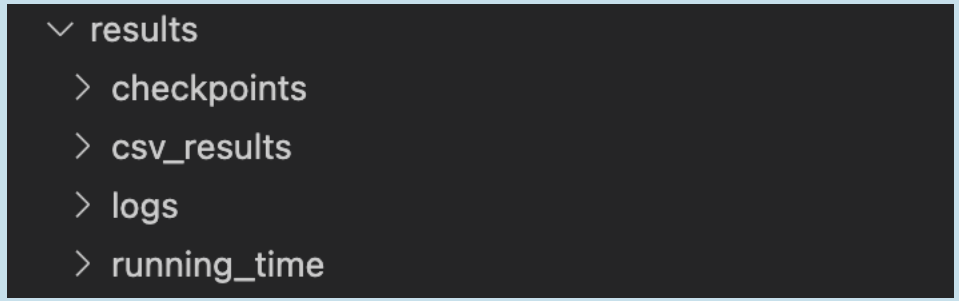


1. checkpoints
2. csv_results: contains csv files that shows training options and each epoch number, train accuracy, test accuracy
3. logs
4. running_time

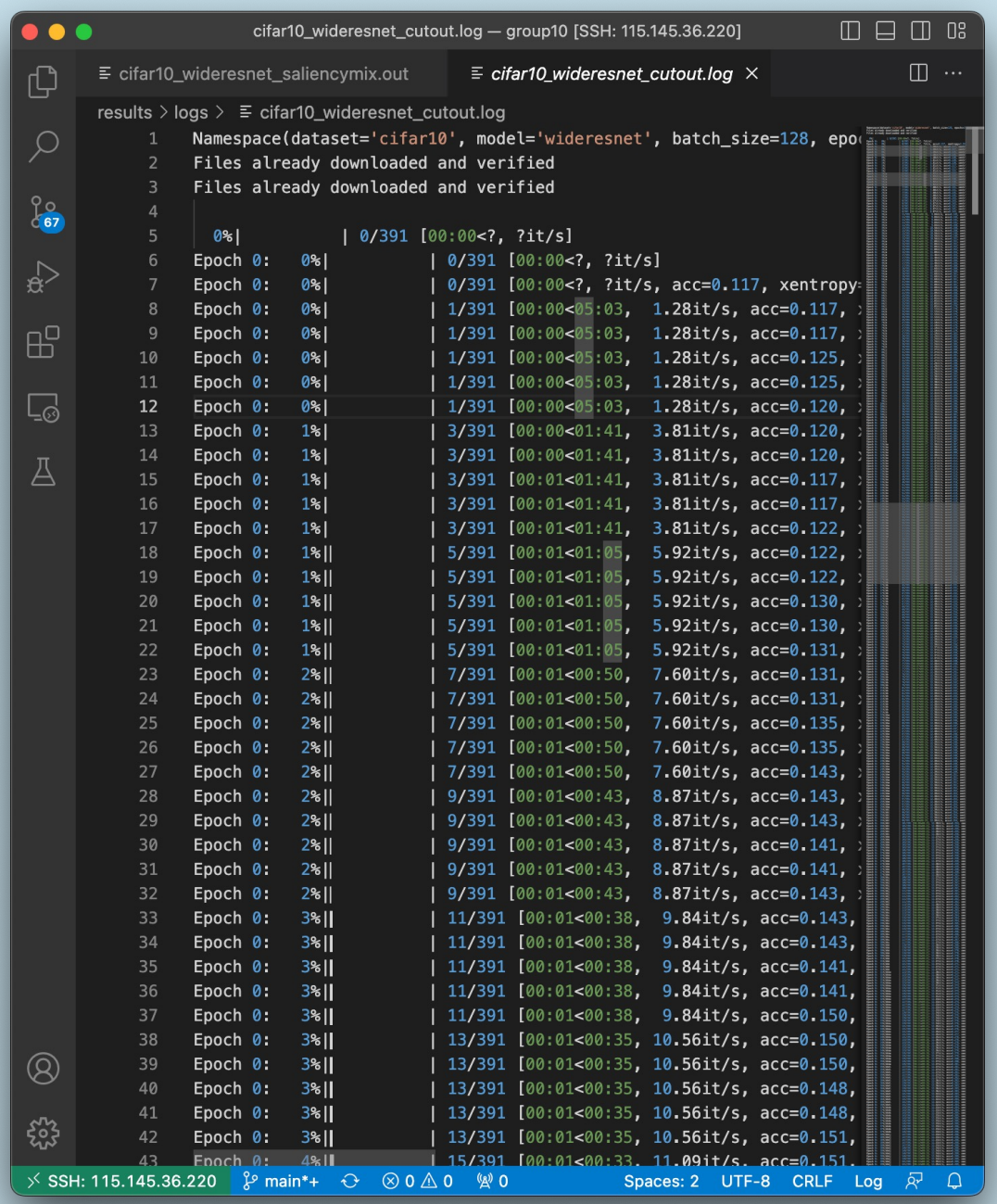


How to see the results? (3)

Our results are divided into 4 directories in ‘results’ directory.

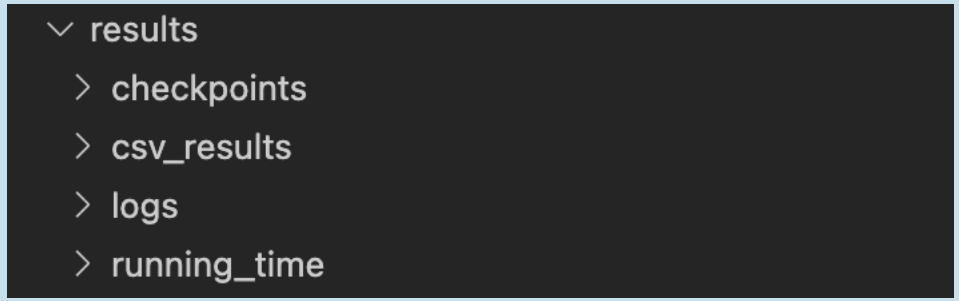


1. checkpoints
2. csv_results
3. **logs** : contains ’.log’ files where you can check training options, and tqdm’s printings
4. running_time

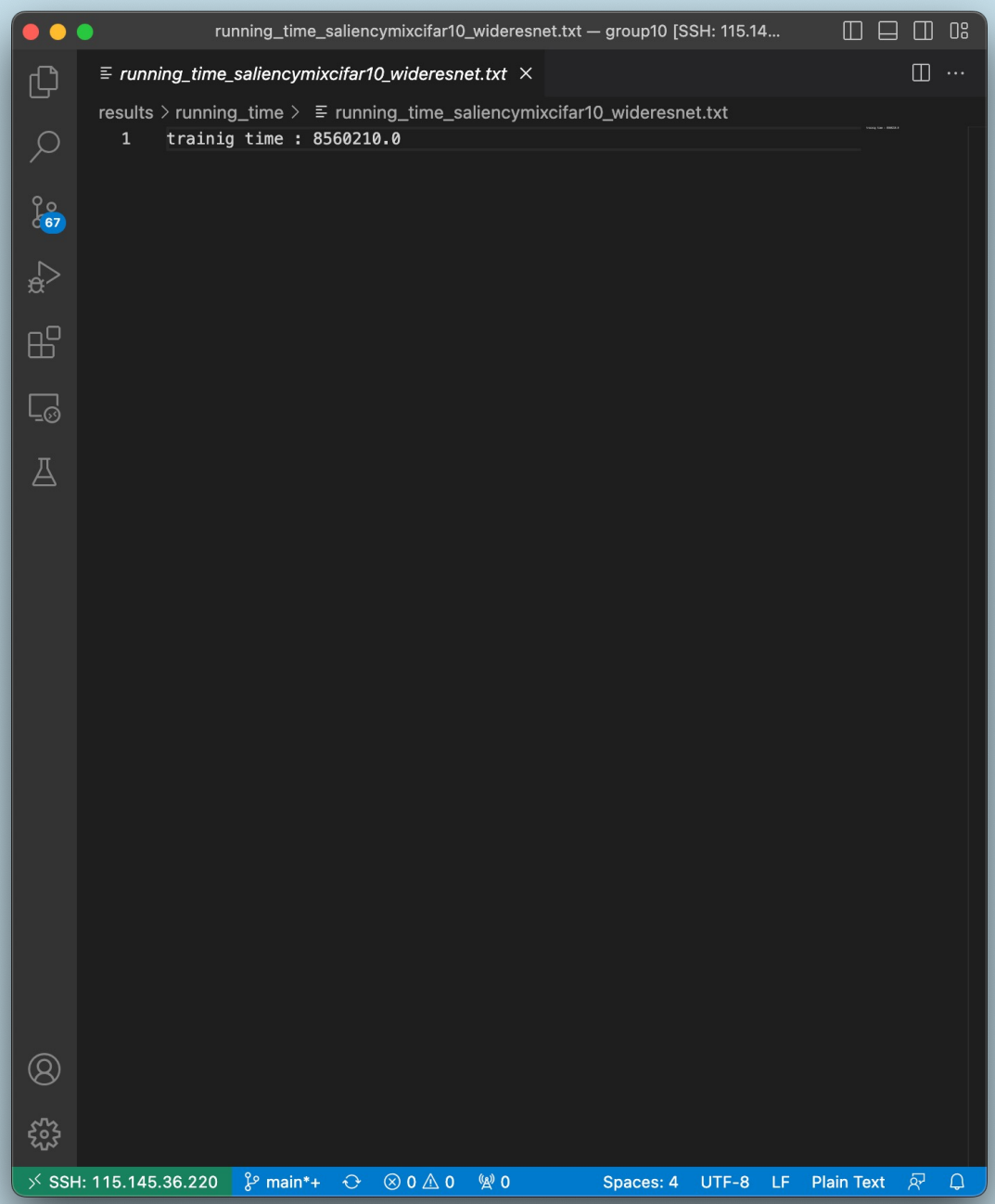


How to see the results? (4)

Our results are divided into 4 directories in ‘results’ directory.

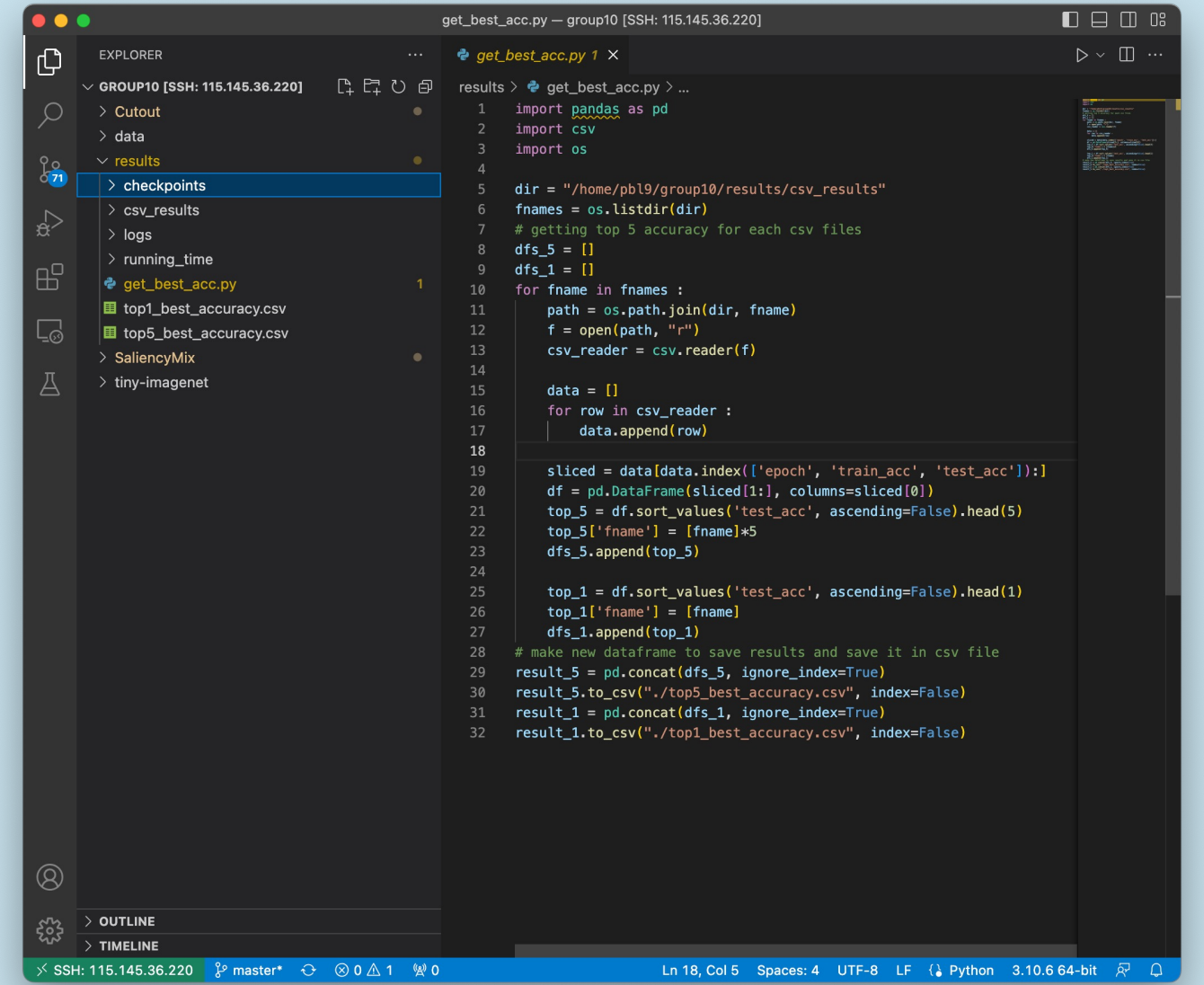


1. checkpoints
2. csv_results
3. logs
4. running_time
   : contains calculated training time of each model



How to see the results? (5)

results/get_best_acc.py



Custom code to get the best accuracy of each model from training results(csv files)