

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

---

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

# Contents

1      **Introduction**

2      **Methods**

3      **Evaluation**

4      **Code demo**

5      **Challenges**

6      **Conclusion**

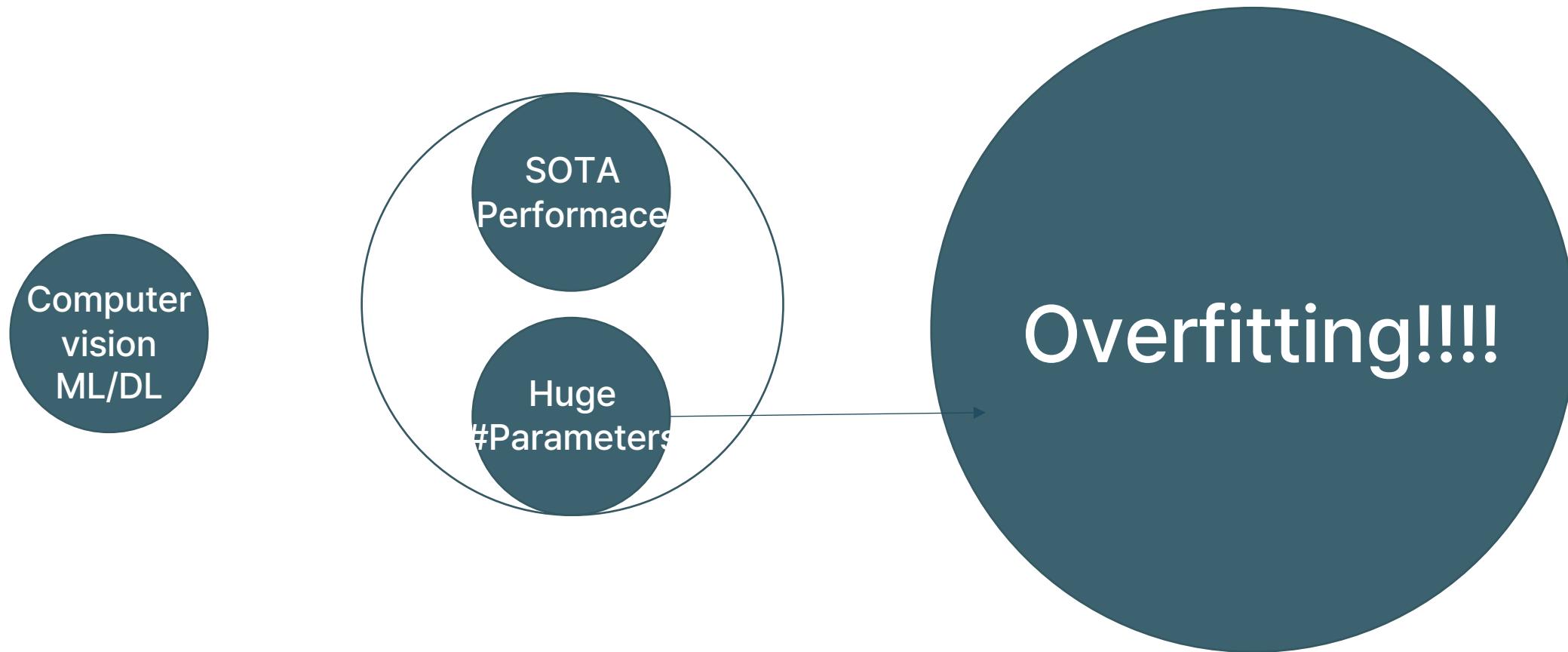


# 1

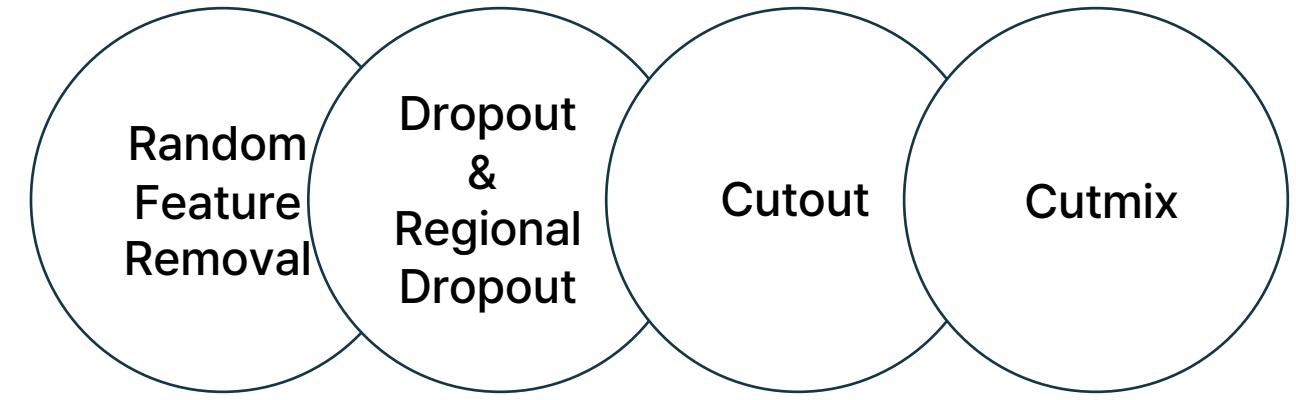
## Introduction & Background

---

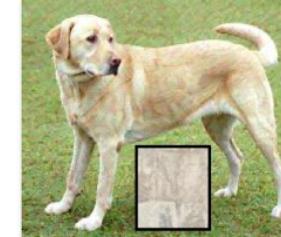
## Introduction



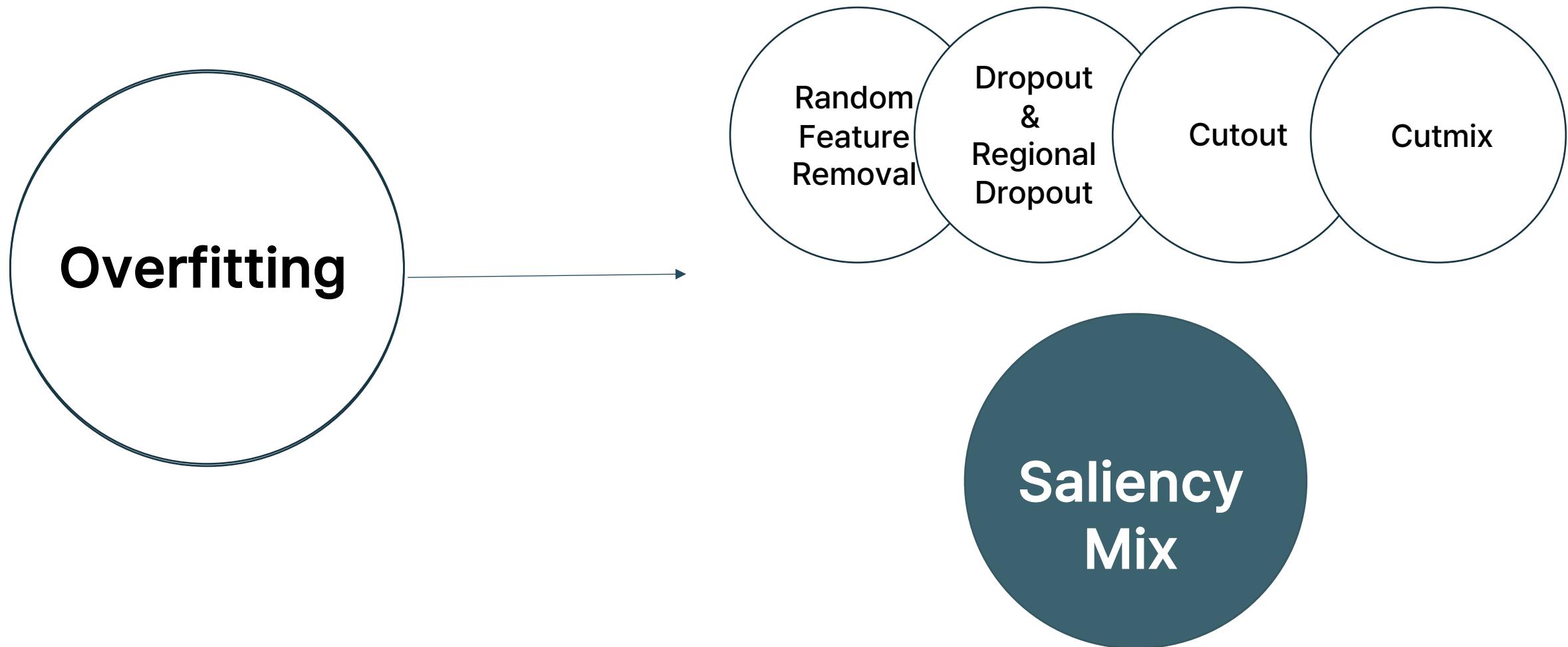
# Data Augmentation



Don't care about informative pixel

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

# Data Augmentation



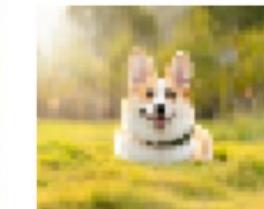
# 2

## Methods

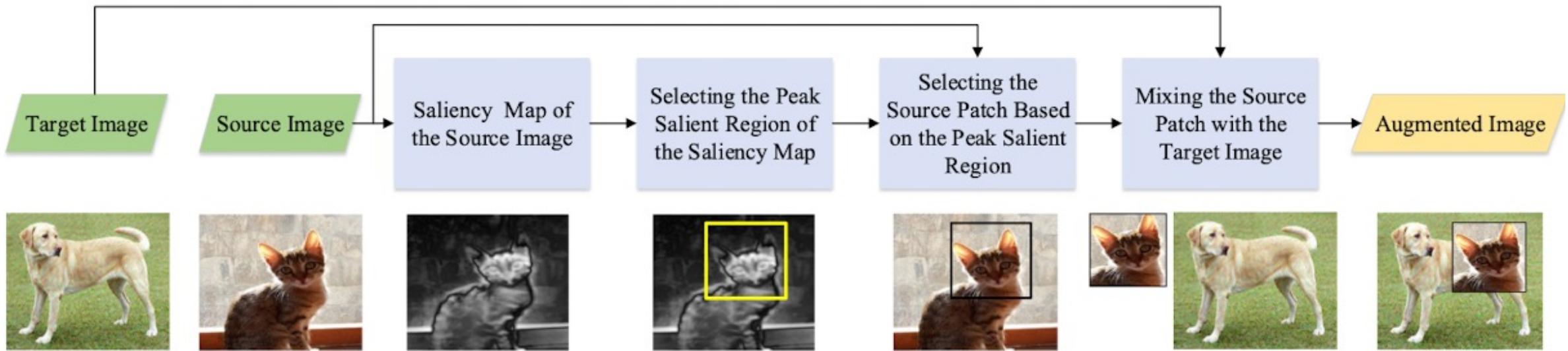
---

# Data augmentation

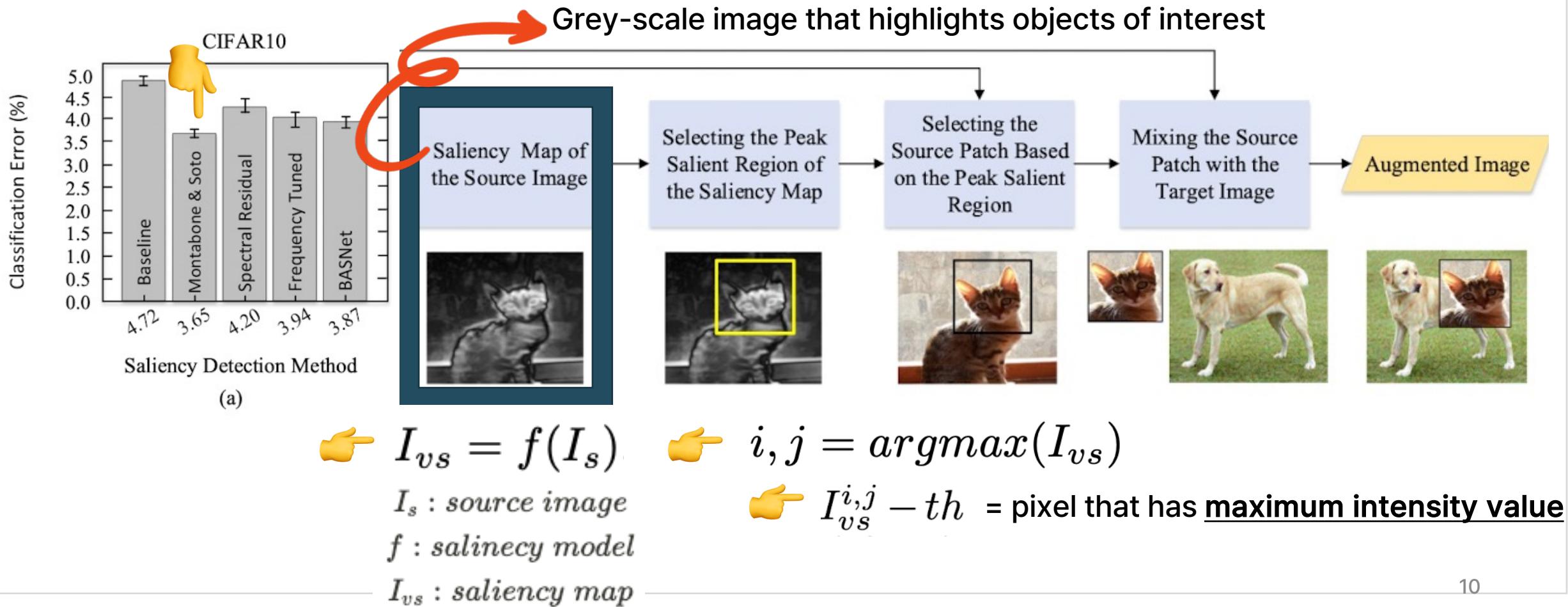
: techniques to artificially increase the amount of data by generating new data points from existing data



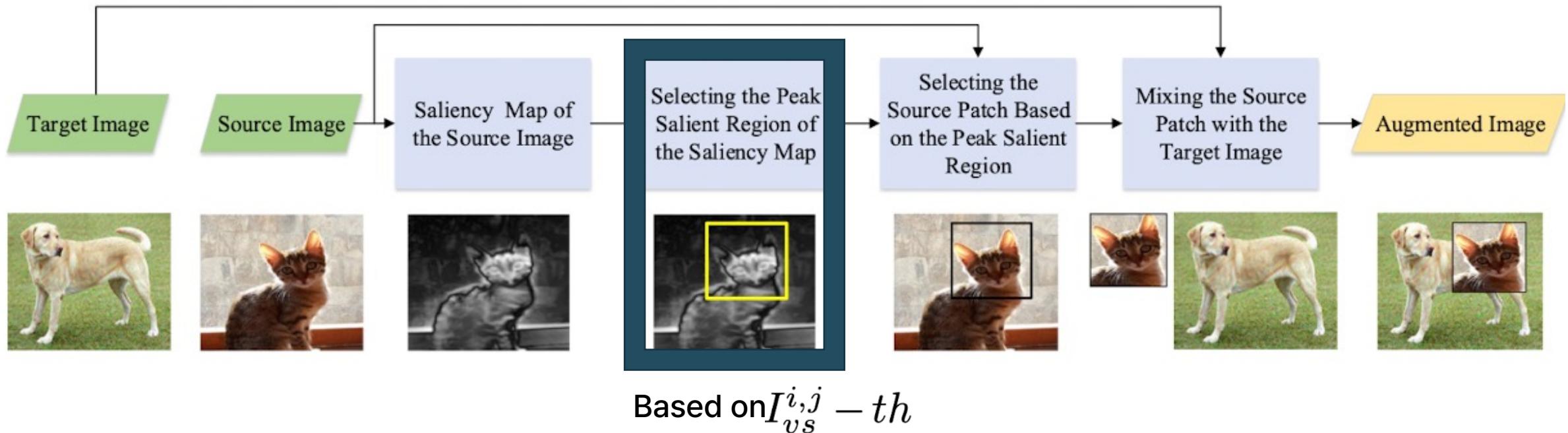
## Process of Saliency Mix data augmentation



## Process of Saliency Mix data augmentation

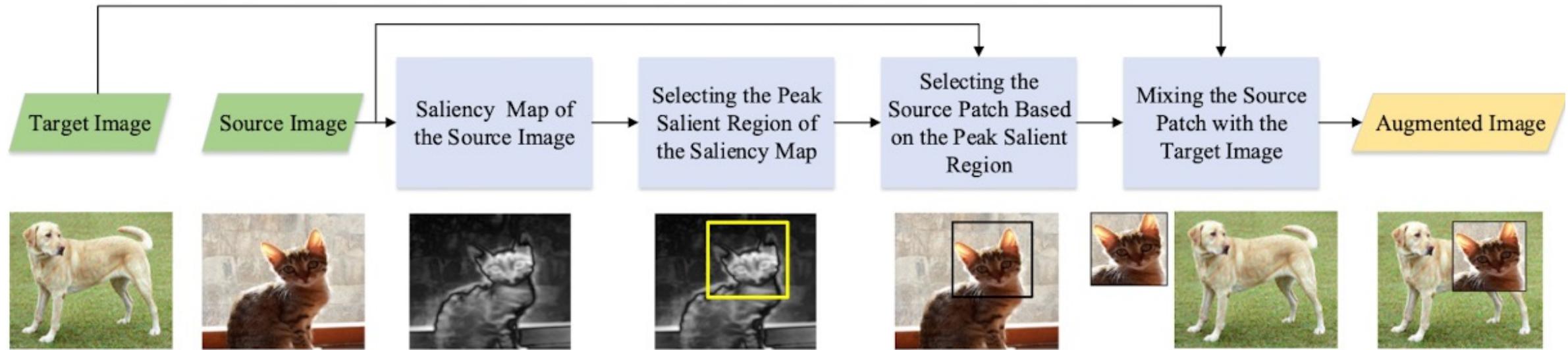
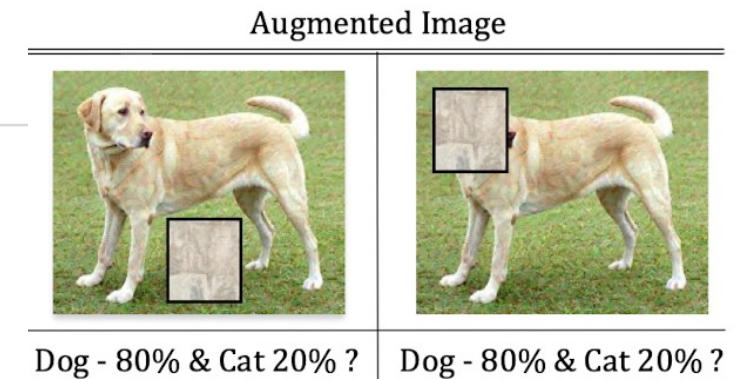


## Process of Saliency Mix data augmentation



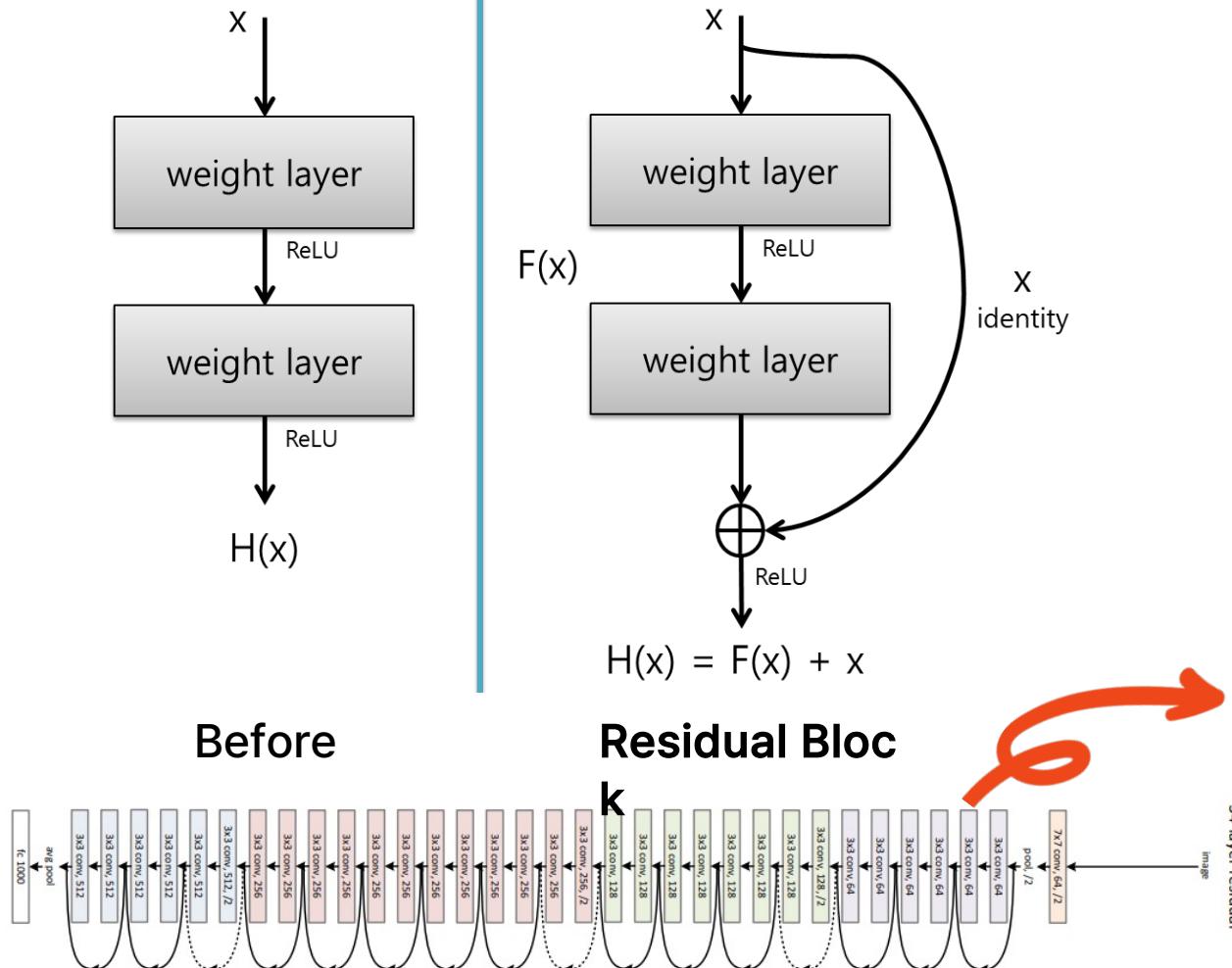
# Method

## Process of Saliency Mix data augmentation



👉 Proper feature representation!

# What is ResNet?(Residual neural Network )



👉 deeper layer  
& solve vanishing gradient problem

# 3

## Evaluation

---

# Paper's training methods

## CIFAR-10 & CIFAR-100

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

METHOD	CIFAR-10	TOP-1 ERROR (%)		
		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
<b>RESNET-18 + SALIENCYMIX</b>	<b>7.59±0.22</b>	<b>3.65±0.10</b>	<b>28.73±0.13</b>	<b>19.29±0.21</b>
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
<b>RESNET-50 + SALIENCYMIX</b>	<b>6.81±0.30</b>	<b>3.46±0.08</b>	<b>24.89±0.39</b>	<b>18.57±0.29</b>
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	-	<b>2.60±0.10</b>	-	17.10±0.30
WIDERESNET-28-10 + PUZZLEMX (200 EPOCHS)	-	-	-	<b>16.23</b>
WIDERESNET-28-10 + CUTMIX	5.18±0.20	2.87±0.16	23.21±0.20	16.66±0.20
<b>WIDERESNET-28-10 + SALIENCYMIX</b>	<b>4.04±0.13</b>	2.76±0.07	<b>19.45±0.32</b>	16.56±0.17



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

## 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 + DROBLOCK	21.87	5.98
RESNET-50 + CUTMIX	21.40	5.92
RESNET-50 + PUZZLEMX	<b>21.24</b>	<b>5.71</b>
<b>RESNET-50 + SALIENCYMIX</b>	<b>21.26</b>	<b>5.76</b>
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
<b>RESNET-101 + SALIENCYMIX</b>	<b>20.09</b>	<b>5.15</b>

# 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

# CIFAR10

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

airplane



automobile



bird



cat



deer



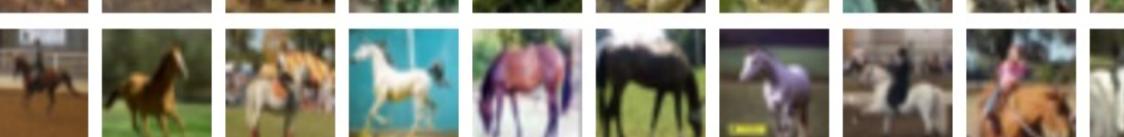
dog



frog



horse



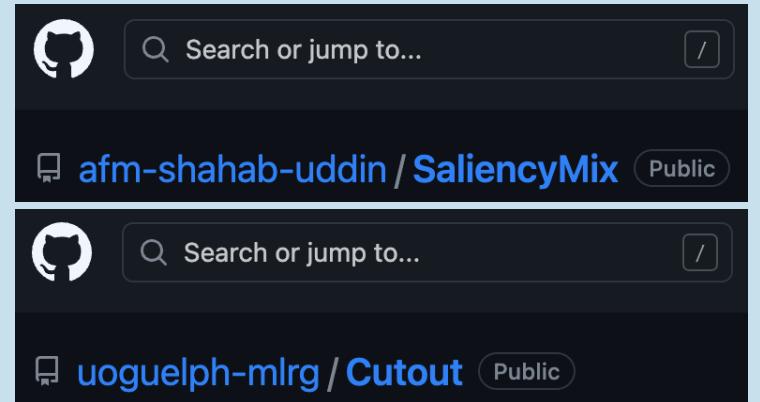
ship



truck

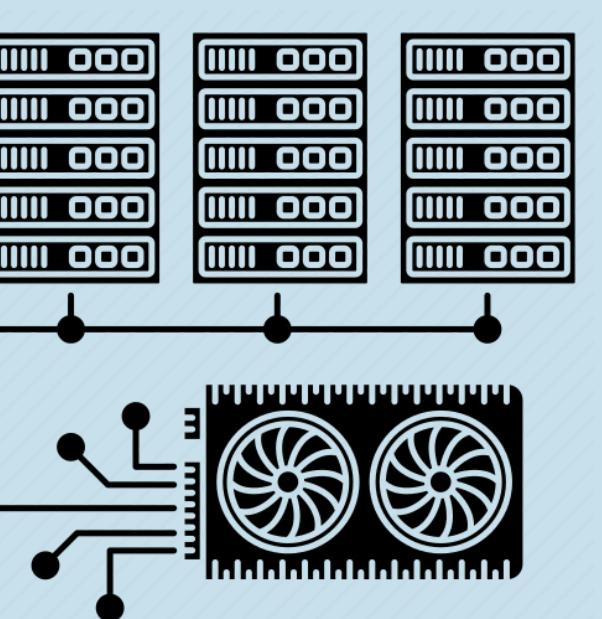


# Training model



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 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

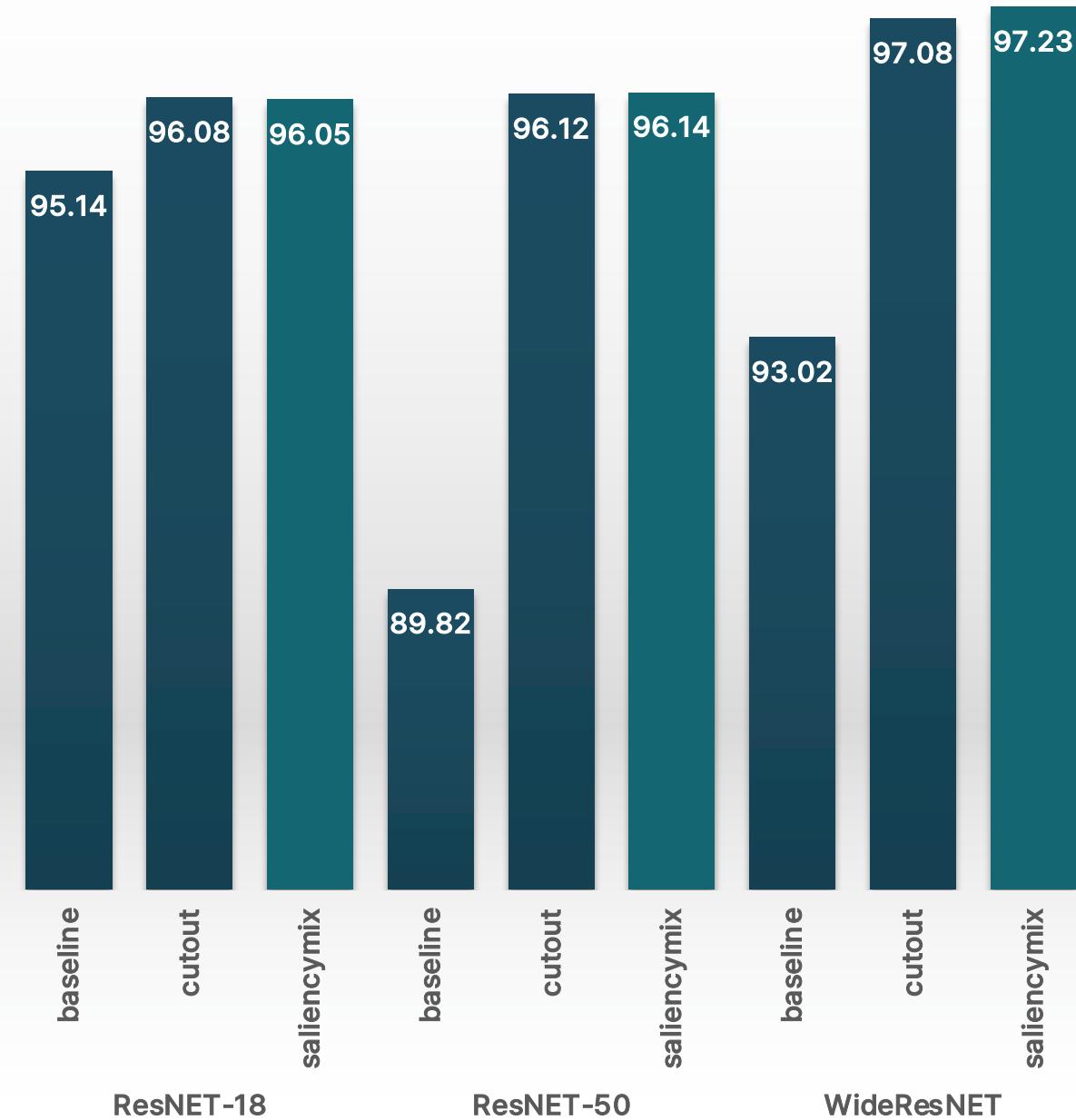
```

# 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

Top-1 Accuracy(%)



# 4

## Code Demo

---

# 5

## Challeneges

---

# 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_DEVICES` command

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

# 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**

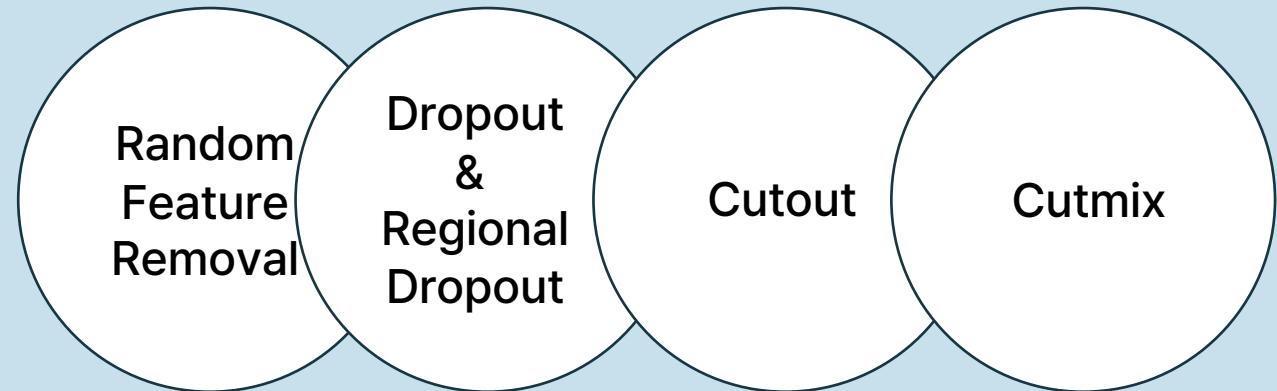
- Augmentation methods not always guarantee better performance
- The basic process randomly selecting and mixing images is the same
- Performance variation present
  
- ✓ **Faster than CutOut**
- ✓ **Meaningful to reduce the variation of performance (loss of information) that can occur in CutOut.**

# 6

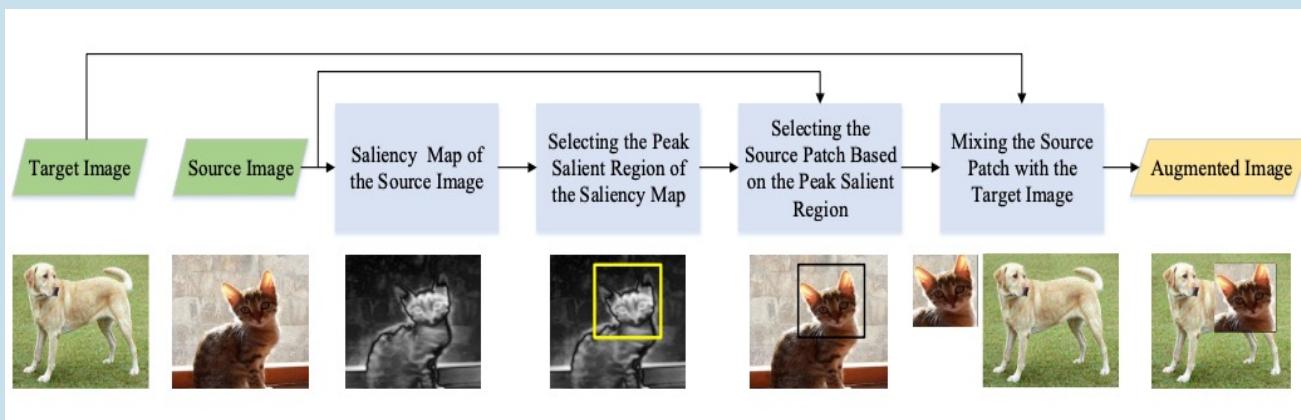
## Conclusion

---

# Summary



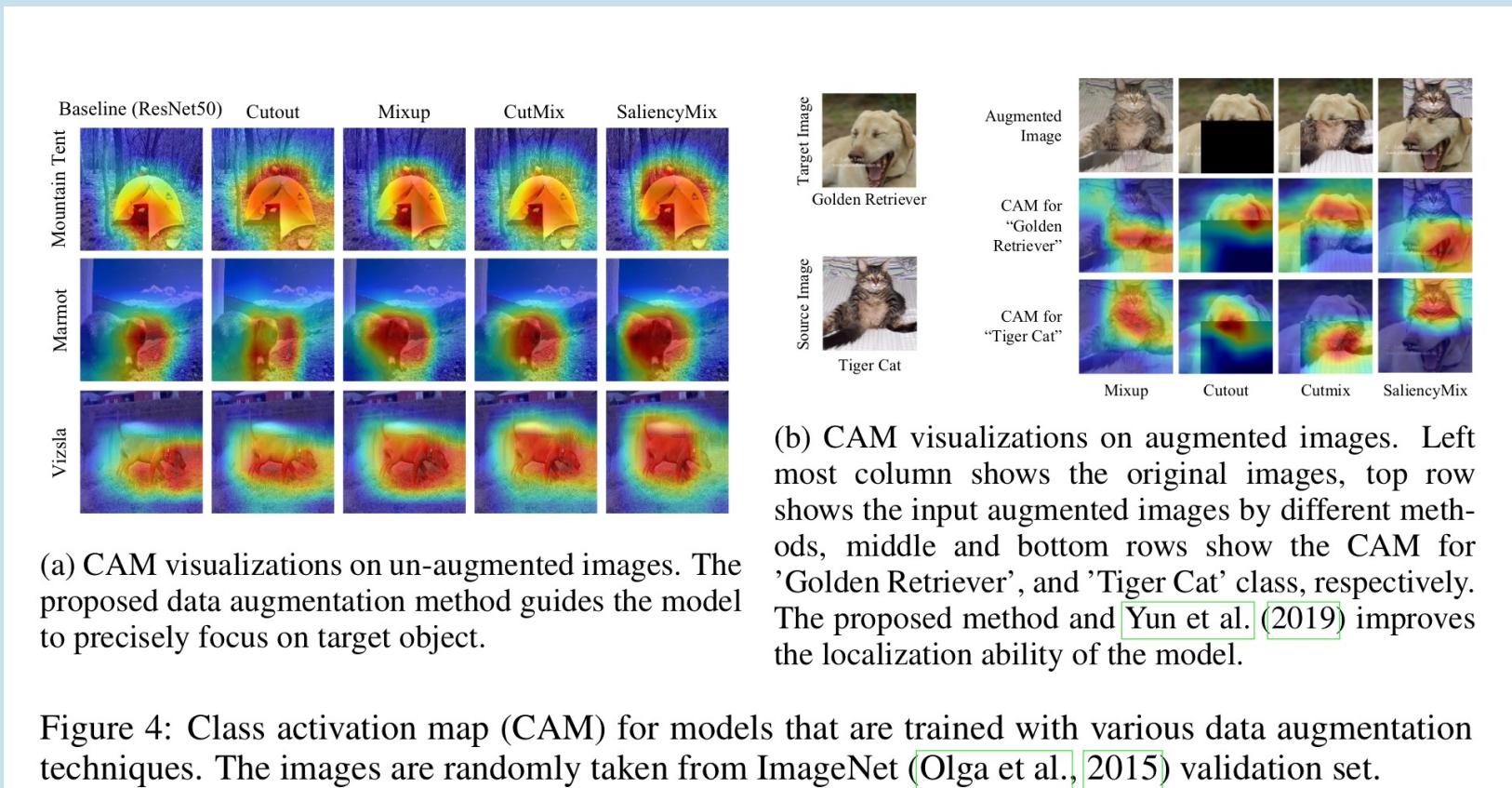
👉 Don't care about informative pixel



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



**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	$4.72 \pm 0.21$
	CutOut	96.08	3.92	$3.99 \pm 0.13$
	SaliencyMix	96.05	3.95	$3.65 \pm 0.10$
ResNET-50	Baseline	95.46	4.54	$4.98 \pm 0.14$
	CutOut	96.12	3.88	$3.86 \pm 0.25$
	SaliencyMix	96.14	3.86	$3.46 \pm 0.08$
WideResNET	Baseline	96.28	3.72	$3.87 \pm 0.08$
	CutOut	97.08	2.92	$3.08 \pm 0.16$
	SaliencyMix	97.23	2.77	$2.76 \pm 0.07$

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

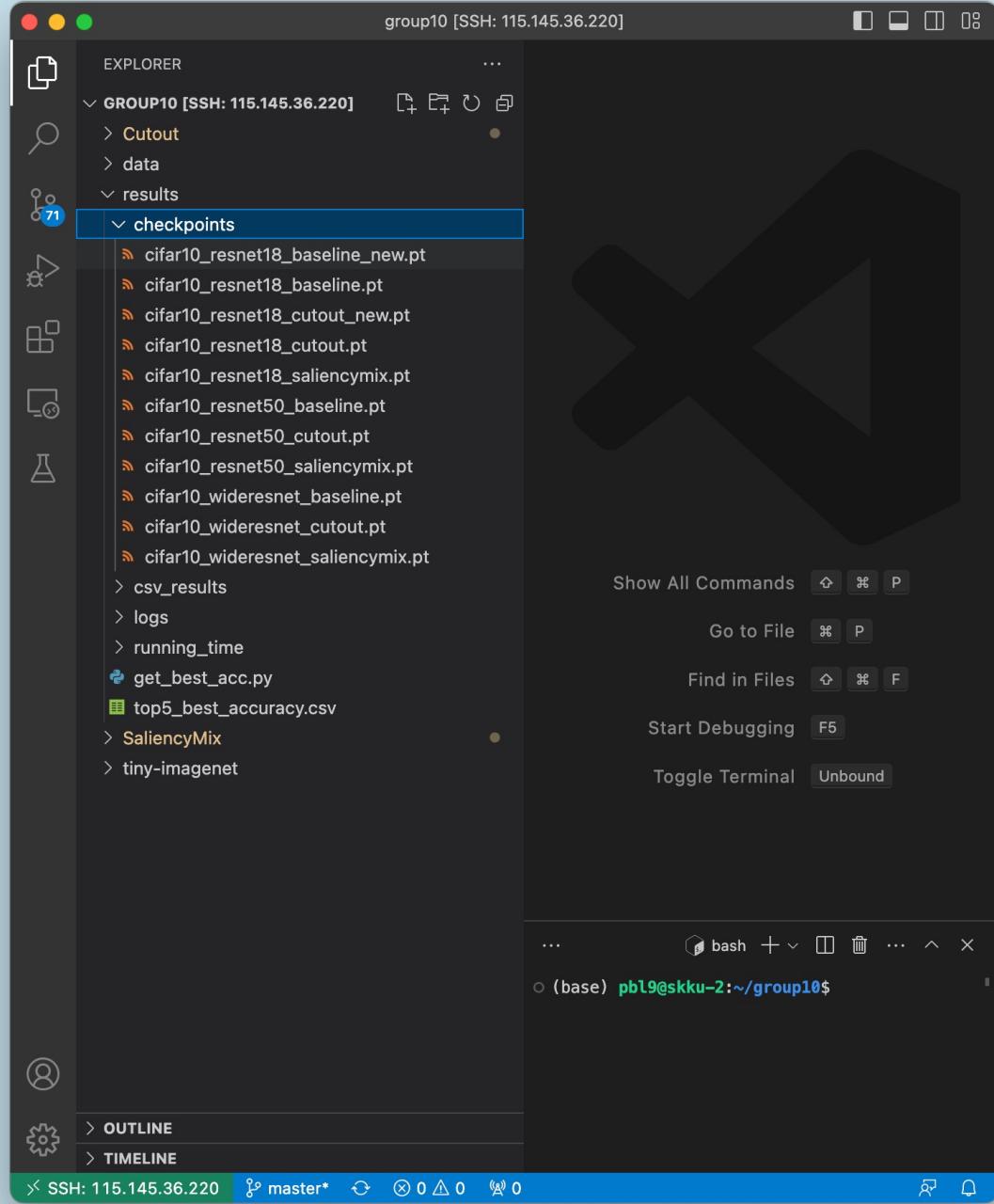
# How to see the results? (1)

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

- ✓ **results**
  - > checkpoints
  - > csv\_results
  - > logs
  - > running\_time

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.

```
    \ results
        > checkpoints
        > csv_results
        > logs
        > running_time
```

- 1. checkpoints
  - 2. **CSV\_res**
  - 3. logs
  - 4. running time

**2. csv\_results**: contains csv files that shows training options and  
3. logs each epoch number, train accuracy, test accuracy

3. logs each epoch number, train accuracy, test accuracy

## 4. running time

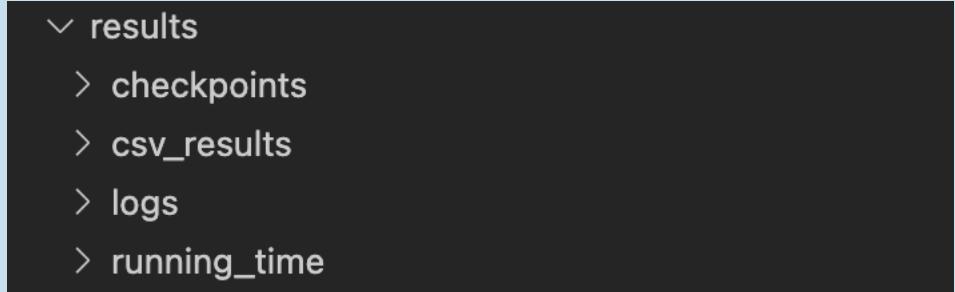
cifar10\_wideresnet\_saliencymix.csv — group10 [SSH: 115.145.36.220]

results > csv\_results > cifar10\_wideresnet\_saliencymix.csv

```
1 dataset,cifar10
2 model,wideresnet
3 batch_size,128
4 epochs,200
5 learning_rate,0.1
6 data_augmentation,True
7 no_cuda,False
8 seed,0
9 beta,1.0
10 salmix_prob,0.5
11 cuda,True
12 """
13 epoch,train_acc,test_acc
14 0,35.128,53.51000000000005
15 1,52.70599999999996,63.12
16 2,59.644,65.96
17 3,63.61599999999999,72.47
18 4,65.628,73.2
19 5,67.984,71.37
20 6,67.46600000000001,73.97
21 7,70.916,81.22
22 8,70.22,81.91000000000001
23 9,71.73,76.77000000000001
24 10,70.798,81.81
25 11,70.906,83.45
26 12,72.32,77.59
27 13,71.846,81.42
28 14,74.078,81.69
29 15,71.678,83.99
30 16,72.688,80.27
31 17,72.13199999999999,84.31
32 18,75.832,80.38
33 19,73.548,83.05
34 20,74.82600000000001,84.57000000000001
35 21,75.14,82.3
36 22,72.886,85.2
37 23,74.24,84.97
38 24,76.328,84.98
39 25,76.59,85.08
40 26,76.254,80.22
41 27,74.9,84.54
42 28,73.9,81.70000000000001
43 29,75.47399999999999,87.53999999999999
```

# 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,
4. running\_time and tqdm's printings

A terminal window showing the 'cifar10\_wideresnet\_cutout.log' file. The log file contains training statistics for a WideresNet model on the CIFAR-10 dataset. It shows epoch progress, batch sizes, and accuracy values. The log file is 67 lines long.

```
cifar10_wideresnet_cutout.log -- group10 [SSH: 115.145.36.220]
cifar10_wideresnet_saliencymix.out  cifar10_wideresnet_cutout.log x

results > logs >  cifar10_wideresnet_cutout.log
1 Namespace(dataset='cifar10', model='wideresnet', batch_size=128, epochs=100)
2 Files already downloaded and verified
3 Files already downloaded and verified
4
5   0%|          | 0/391 [00:00<?, ?it/s]
6 Epoch 0:  0%|          | 0/391 [00:00<?, ?it/s]
7 Epoch 0:  0%|          | 0/391 [00:00<?, ?it/s, acc=0.117, xentropy=0.883]
8 Epoch 0:  0%|          | 1/391 [00:00<05:03, 1.28it/s, acc=0.117, xentropy=0.883]
9 Epoch 0:  0%|          | 1/391 [00:00<05:03, 1.28it/s, acc=0.117, xentropy=0.883]
10 Epoch 0:  0%|          | 1/391 [00:00<05:03, 1.28it/s, acc=0.125, xentropy=0.883]
11 Epoch 0:  0%|          | 1/391 [00:00<05:03, 1.28it/s, acc=0.125, xentropy=0.883]
12 Epoch 0:  0%|          | 1/391 [00:00<05:03, 1.28it/s, acc=0.120, xentropy=0.883]
13 Epoch 0:  1%|          | 3/391 [00:00<01:41, 3.81it/s, acc=0.120, xentropy=0.883]
14 Epoch 0:  1%|          | 3/391 [00:00<01:41, 3.81it/s, acc=0.120, xentropy=0.883]
15 Epoch 0:  1%|          | 3/391 [00:01<01:41, 3.81it/s, acc=0.117, xentropy=0.883]
16 Epoch 0:  1%|          | 3/391 [00:01<01:41, 3.81it/s, acc=0.117, xentropy=0.883]
17 Epoch 0:  1%|          | 3/391 [00:01<01:41, 3.81it/s, acc=0.122, xentropy=0.883]
18 Epoch 0:  1%||         | 5/391 [00:01<01:05, 5.92it/s, acc=0.122, xentropy=0.883]
19 Epoch 0:  1%||         | 5/391 [00:01<01:05, 5.92it/s, acc=0.122, xentropy=0.883]
20 Epoch 0:  1%||         | 5/391 [00:01<01:05, 5.92it/s, acc=0.130, xentropy=0.883]
21 Epoch 0:  1%||         | 5/391 [00:01<01:05, 5.92it/s, acc=0.130, xentropy=0.883]
22 Epoch 0:  1%||         | 5/391 [00:01<01:05, 5.92it/s, acc=0.131, xentropy=0.883]
23 Epoch 0:  2%||         | 7/391 [00:01<00:50, 7.60it/s, acc=0.131, xentropy=0.883]
24 Epoch 0:  2%||         | 7/391 [00:01<00:50, 7.60it/s, acc=0.131, xentropy=0.883]
25 Epoch 0:  2%||         | 7/391 [00:01<00:50, 7.60it/s, acc=0.135, xentropy=0.883]
26 Epoch 0:  2%||         | 7/391 [00:01<00:50, 7.60it/s, acc=0.135, xentropy=0.883]
27 Epoch 0:  2%||         | 7/391 [00:01<00:50, 7.60it/s, acc=0.143, xentropy=0.883]
28 Epoch 0:  2%||         | 9/391 [00:01<00:43, 8.87it/s, acc=0.143, xentropy=0.883]
29 Epoch 0:  2%||         | 9/391 [00:01<00:43, 8.87it/s, acc=0.143, xentropy=0.883]
30 Epoch 0:  2%||         | 9/391 [00:01<00:43, 8.87it/s, acc=0.141, xentropy=0.883]
31 Epoch 0:  2%||         | 9/391 [00:01<00:43, 8.87it/s, acc=0.141, xentropy=0.883]
32 Epoch 0:  2%||         | 9/391 [00:01<00:43, 8.87it/s, acc=0.143, xentropy=0.883]
33 Epoch 0:  3%||         | 11/391 [00:01<00:38, 9.84it/s, acc=0.143, xentropy=0.883]
34 Epoch 0:  3%||         | 11/391 [00:01<00:38, 9.84it/s, acc=0.143, xentropy=0.883]
35 Epoch 0:  3%||         | 11/391 [00:01<00:38, 9.84it/s, acc=0.141, xentropy=0.883]
36 Epoch 0:  3%||         | 11/391 [00:01<00:38, 9.84it/s, acc=0.141, xentropy=0.883]
37 Epoch 0:  3%||         | 11/391 [00:01<00:38, 9.84it/s, acc=0.150, xentropy=0.883]
38 Epoch 0:  3%||         | 13/391 [00:01<00:35, 10.56it/s, acc=0.150, xentropy=0.883]
39 Epoch 0:  3%||         | 13/391 [00:01<00:35, 10.56it/s, acc=0.150, xentropy=0.883]
40 Epoch 0:  3%||         | 13/391 [00:01<00:35, 10.56it/s, acc=0.148, xentropy=0.883]
41 Epoch 0:  3%||         | 13/391 [00:01<00:35, 10.56it/s, acc=0.148, xentropy=0.883]
42 Epoch 0:  3%||         | 13/391 [00:01<00:35, 10.56it/s, acc=0.151, xentropy=0.883]
43 Epoch 0:  4%||         | 15/391 [00:01<00:33, 11.09it/s, acc=0.151, xentropy=0.883]
```

# How to see the results? (4)

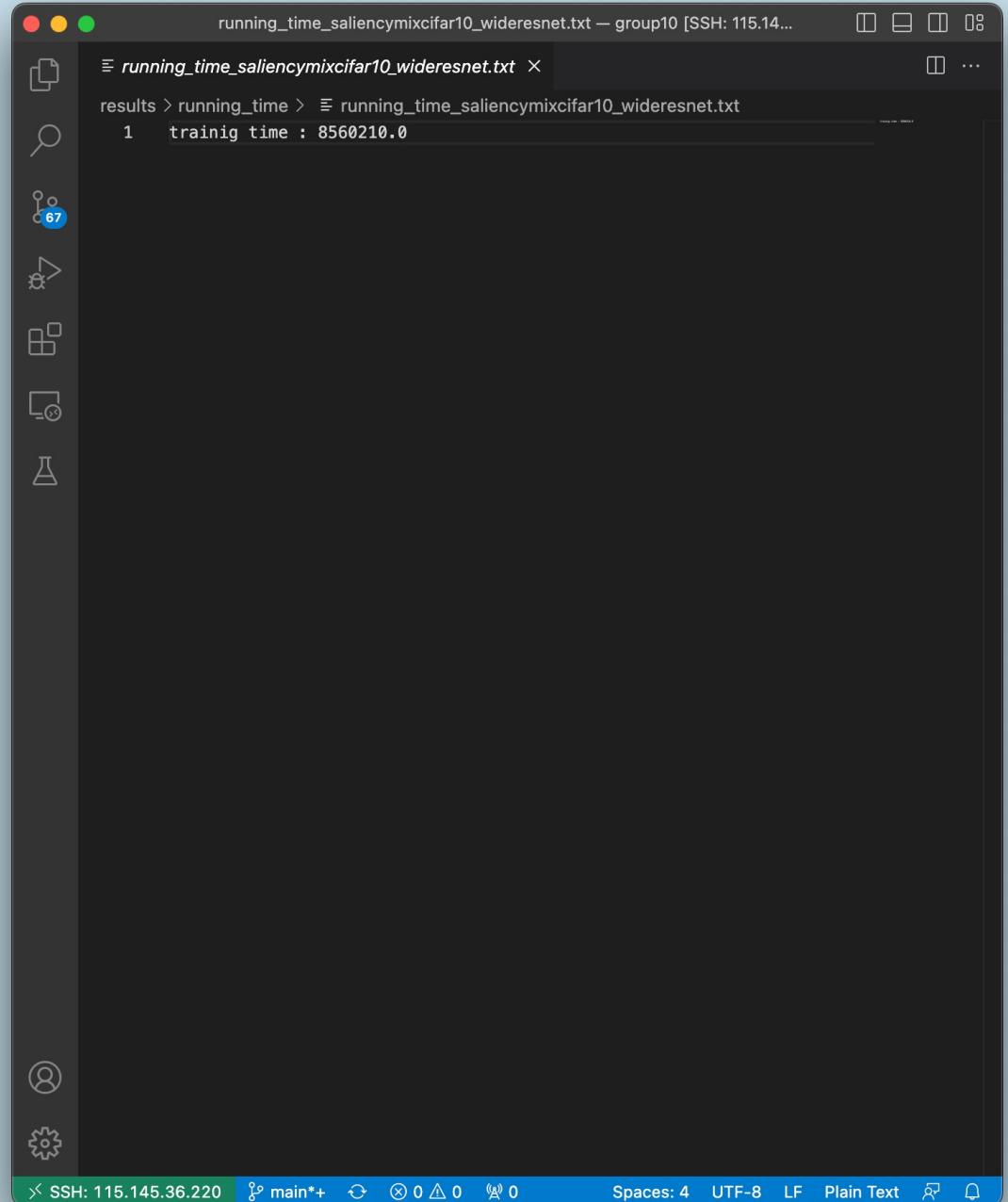
Our results are divided into 4 directories in ‘results’ directory.

```
✓ results
  > checkpoints
  > csv_results
  > logs
  > running_time
```

1. checkpoints
2. csv\_results
3. logs

## 4. running\_time

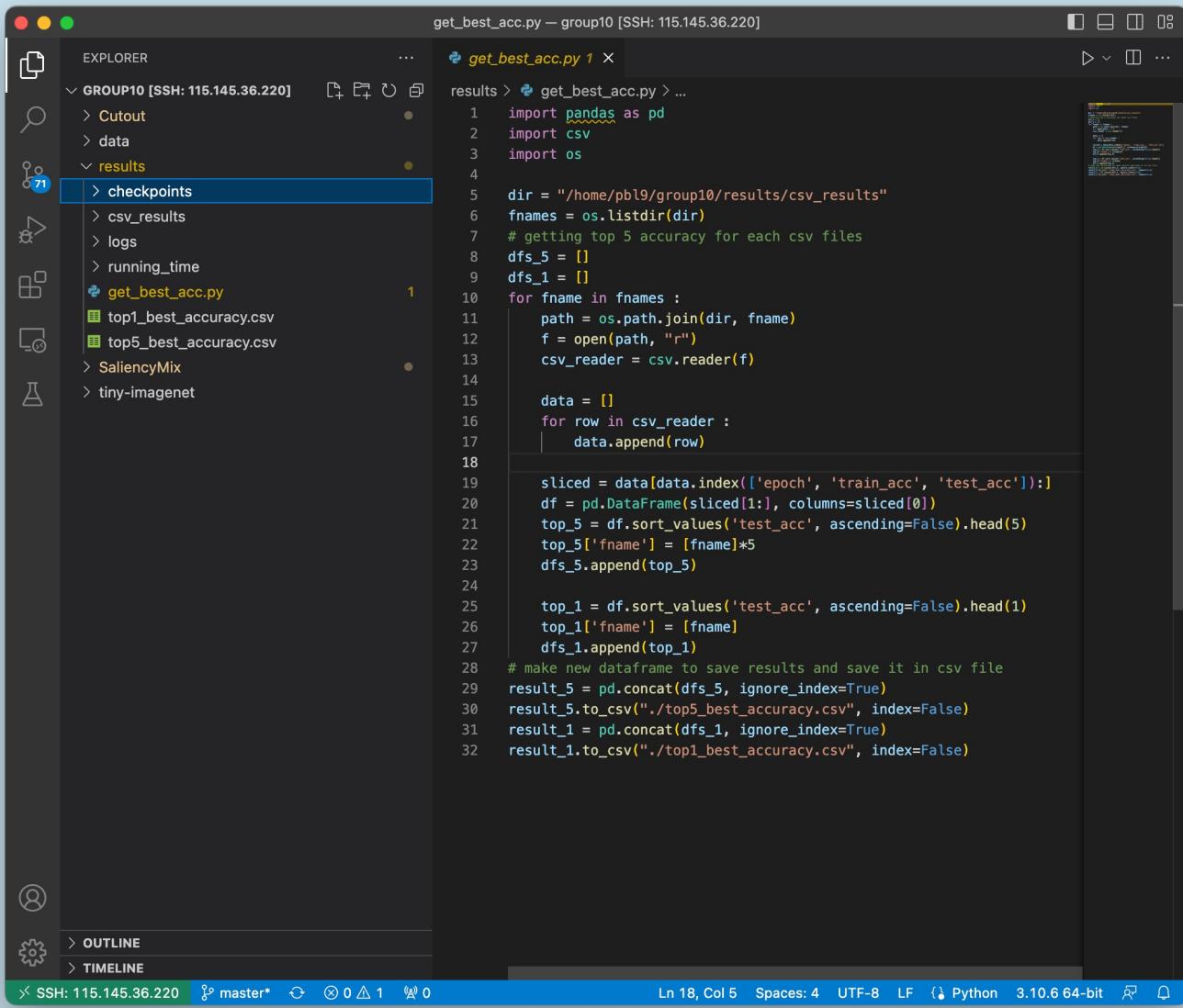
: contains calculated training time of each model



```
running_time_saliencymixcifar10_wideresnet.txt — group10 [SSH: 115.14...]
running_time_saliencymixcifar10_wideresnet.txt
results > running_time > running_time_saliencymixcifar10_wideresnet.txt
1 trainig time : 8560210.0
```

# How to see the results? (5)

results/get\_best\_acc.py



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** get\_best\_acc.py — group10 [SSH: 115.145.36.220]
- Left Sidebar (EXPLORER):** Shows a file tree under GROUP10 [SSH: 115.145.36.220]. The "results" folder is expanded, showing "checkpoints" (selected), "csv\_results", "logs", "running\_time", "get\_best\_acc.py", "top1\_best\_accuracy.csv", "top5\_best\_accuracy.csv", "SaliencyMix", and "tiny-imagenet".
- Code Editor:** The code for "get\_best\_acc.py" is displayed. The code reads CSV files from a directory, extracts top 5 accuracy rows for each, and saves them to "top5\_best\_accuracy.csv" and "top1\_best\_accuracy.csv".

```
import pandas as pd
import csv
import os

dir = "/home/pbl9/group10/results/csv_results"
fnames = os.listdir(dir)
# getting top 5 accuracy for each csv files
dfs_5 = []
dfs_1 = []

for fname in fnames :
    path = os.path.join(dir, fname)
    f = open(path, "r")
    csv_reader = csv.reader(f)

    data = []
    for row in csv_reader :
        data.append(row)

    sliced = data[data.index(['epoch', 'train_acc', 'test_acc']):]
    df = pd.DataFrame(sliced[1:], columns=sliced[0])
    top_5 = df.sort_values('test_acc', ascending=False).head(5)
    top_5['fname'] = [fname]*5
    dfs_5.append(top_5)

    top_1 = df.sort_values('test_acc', ascending=False).head(1)
    top_1['fname'] = [fname]
    dfs_1.append(top_1)

# make new dataframe to save results and save it in csv file
result_5 = pd.concat(dfs_5, ignore_index=True)
result_5.to_csv("./top5_best_accuracy.csv", index=False)
result_1 = pd.concat(dfs_1, ignore_index=True)
result_1.to_csv("./top1_best_accuracy.csv", index=False)
```
- Bottom Status Bar:** Shows the connection status (SSH: 115.145.36.220), branch (master\*), and other system information.

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