FlexiFed-IMP

Implementation of paper - FlexiFed: Personalized Federated Learning for Edge Clients with Heterogeneous Model Architectures

Background

Federated Learning (FL) allows edge mobile and Web-of-Things devices to train a shared global Machine Learning (ML) model under the orchestration of a center parameter server, which has tackled the challenge of privacy issue and performance issue. But the existing FL schemes are mainly concerned with data heterogeneity and they cannot accommodate architecture heterogeneity - clients running ML models with different architectures cannot train a ML model collaboratively under these schemes.

In this paper, experiments are conducted with four widely-used ML models on four public datasets, which demonstrate the usefulness of FlexiFed. There are three aggregation strategies under the frame of the FlexiFed and I will present the processes and results of Implementation of the paper in this repository.

Requirement

- Python 3.8 (ubuntu20.04)
- Pytorch 2.0.0
- Cuda 11.8
- GPU RTX 4090
- Necessary lib

```
pip install numpy
pip install pandas
pip install librosa==0.9.1
```

The environment for running the project is provided by AutoDL.

Structure

The structure of the project is shown below :

```
├─src
|   ├─model
|   |   ├─VGG.py
|   |   ├─ResNet.py
|   |   ├─CharCNN.py
|   |   └─VDCNN.py
|   |
|   ├─utils
|   |   ├─Dataset.py
|   |   └─Visualization.py
|   |
|   └─core
```

```
|     ├─Client.py
|     ├─FlexiFed.py
|     └─main.py
|
├─dataset
├─model
├─result
├─final result
├─images
|
├─materials
|   ├─Main idea.pdf
|   └─Reproduction process.pdf
|
├─README.pdf
└─README.md
```

- src/model :

  - VGG.py : There are four models in VGG-family - VGG-11, VGG-13, VGG-16, VGG-19
  - ResNet.py : There are four models in ResNet-family - ResNet-20, VGG-32, VGG-44, VGG-56
  - CharCNN.py : There are four models in CharCNN-family - CharCNN-3, CharCNN-4, CharCNN-5, CharCNN-6
  - VDCNN.py : There are four models in VDCNN-family - VDCNN-9, VDCNN-17, VDCNN-29, VDCNN-49

  Use the functions provided by each pythonfile to create the model you want, e.g. :

```
model1=vgg11_bn(input_channel=3,w=32,h=32,num_classes=10)
model2=resnet20(input_channel=3,w=32,h=32,num_classes=10)
model3=charcnn3(m=70,l0=1014,num_classes=4)
model4=vdcnn9(m=69,l0=1024,num_classes=4)
```

- src/utils :

  - Dataset.py : There are some necessary functions to get the dataset (`get_dataset`) and split the dataset by index (`get_idx_dict`). The relevant dataset includes : CIFAR-10, CINIC-10, Speech-Commands, AG-News
  - Visualization : There are functions to visualize the result of the FL schemes and two figures will be created : the convergence process of models in the same family with different version under the same aggregation strategy, the convergence process of models in the same family with same version under different aggregation strategy

- src/core:

  - Clients.py : There is an important class to simulate the clients in edge, which has the ability to train and test locally
  - FlexiFed.py : The FlexiFed frame named ParamserServer, which has a client-set to simulate the clients in the FL System, has three aggregation strategies (Basic-Common, Clustered-Common, Max-Common) and a function to train clients globally
  - main.py : The main function to run the FL System, you can run different system by setting the parameters : num_clients (the number of the clients in FL System),

> family_name (the model family), dataset_name, communication_round (the rounds of global training), strategy (the aggregation strategy)

- dataset : You can download the dataset from the link in this folder.
- model : You can download the models from the link in this folder.
- result : There are the .csv files that record the convergence process, includes : VGG-CIFAR-10, VGG-CINIC-10, VGG-Speech-Commands, ResNet-CIFAR-10, ResNet-CINIC-10, ResNet-Speech-Commands, CharCNN-AG-NEWS, VDCNN-AG-NEWS
- final result : There are .docx and .pdf files recording the accuracy table of the final result.
- images : There are relevant images for this project.
- materials :
  - Main idea.pdf : Here is a record of my interpretation of this paper.

Datasets

Four datasets are used to train the model :

- CIFAR-10 : A dataset used for image classification, which has 60,000 32x32 RGB images in 10 classes (train : test = 5 : 1) .

  You can use the pytorch to download the CIFAR-10 dataset directly, like :

  ```
  from torchvision import datasets
  train_dataset =
  datasets.CIFAR10(path+"/train",train=True,download=False,transform=data_tran
  sforms['train'])
  test_dataset =
  datasets.CIFAR10(path+"/test",train=False,download=False,transform=data_tran
  sforms['test'])
  ```

  We use this dataset to train VGG-family and ResNet-family.

- CINIC-10 : A dataset used for image classification, which has 270,000 32x32 RGB images in 10 classes (train : test = 2 : 1) .

  For this dataset, I combine the training samples and validation samples to train the model, you can use this link to download the dataset and merage train-valid locally. To create the dataset in pytorch, you can use the ImageFolder, like :

  ```
  from torchvision import datasets
  train_dataset =
  datasets.ImageFolder(path+"/train",transform=data_transforms['train'])
  test_dataset =
  datasets.ImageFolder(path+"/test",transform=data_transforms['test'])
  ```

  We use this dataset to train VGG-family and ResNet-family.

- Speech-Commands : A dataset used for speech recognition, which has 65,000 one-second-long audio clips of different keywords. We just care about the classes include : 'down', 'go', 'left', 'no', 'off', 'on', 'right', 'stop', 'up', 'yes', so that we will set the label of other classes to '_unknown_'. Note that there is also a class named '_silence_', I split the audio in _background_noise_ to one-second-long clips and add some waveforms initialized by zero to constitute the '_silence_' data in the train dataset.

You can use this link to download the train dataset and use this link to download the test dataset. Note that the version of the Speech-Commands dataset I used is v0.01. You will get a 1x16,000 array after loading an audio and I will transform this array to a 1x32x32 matrix by some complicated operations (you can read the code of `src/utils/Dataset.py` to get some details) . Finally, we can load the dataset by the class we design, like :

```
train_dataset=Speech_CommandsDataset(path+"/train",data_transforms['train'])
test_dataset=Speech_CommandsDataset(path+"/test",data_transforms['test'])
```

We use this dataset to train VGG-family and ResNet-family.

- AG-News : A dataset used for text classification, which has 120,000 training samples and 7,600 testing samples. The data is stored in train.csv and test.csv, we will read the columns to get enough characters for our model.

  You can use this link to download the dataset. Let m be the length of the alphabet of the model, l0 be the number of the characters used by the model (e.g. CharCNN - m=70,l0=1014; VDCNN - m=69,l0=1024) , and we will get a mxl0 matrix after loading the news. You can use the AG_NewsDataset (you can read the code of `src/utils/Dataset.py` to get some details) to load the dataset, like :

  ```
  # CharCNN
  train_dataset=AG_NewsDataset(path+"/train/train.csv",1014,alphabet)
  test_dataset=AG_NewsDataset(path+"/test/test.csv",1014,alphabet)
  # VDCNN
  train_dataset=AG_NewsDataset(path+"/train/train.csv",1024,alphabet)
  test_dataset=AG_NewsDataset(path+"/test/test.csv",1024,alphabet)
  ```

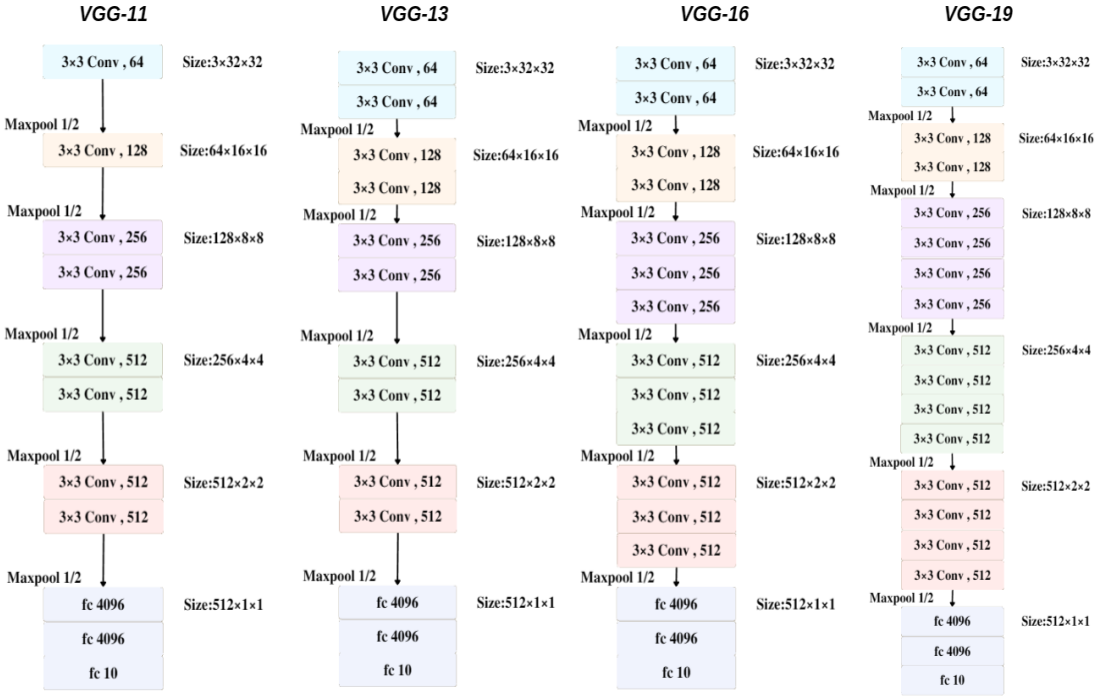
  We use this dataset to train CharCNN-family and VDCNN-family.

Model

Four model familes are used to implement the experiment :

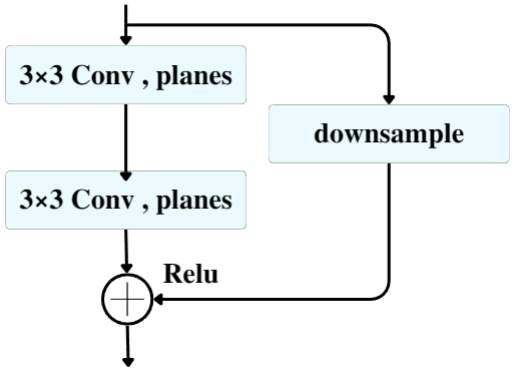
- VGG-family : Visual Geometry Group is an architecture of deep neural network (nn) , it performed very well on ILSVRC 2014 and proved that increasing the depth of the network can affect the final performance of the network to a certain extent.

  We use the four versions of VGG-family : VGG-11, VGG-13, VGG-16, VGG-19, and the architectures of the nn are shown below :

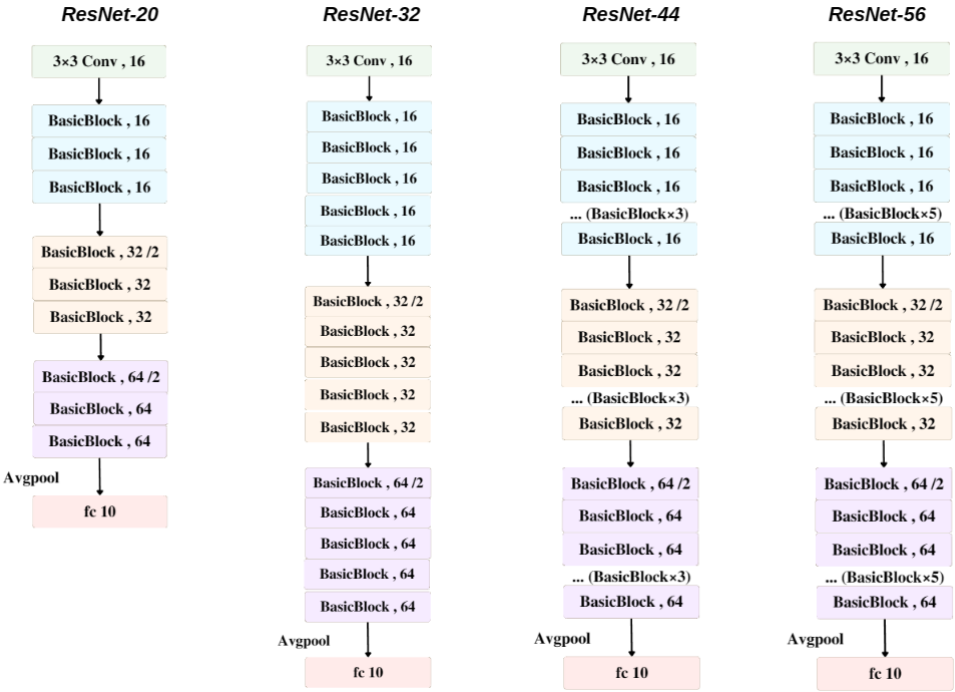


We can see that there is only one common base layer between different versions of VGG-family, so that we can expect that the difference between different aggregation strategies in VGG model may be obvious.

- ResNet-family : ResNet is proposed mainly to tackle the degradation problem in deep nn and realize the proportional relationship between the depth of the network and model accuracy. The nn versions we use are stacks of BasicBlock, whose architecture is shown below (BasicBlock(planes)) :



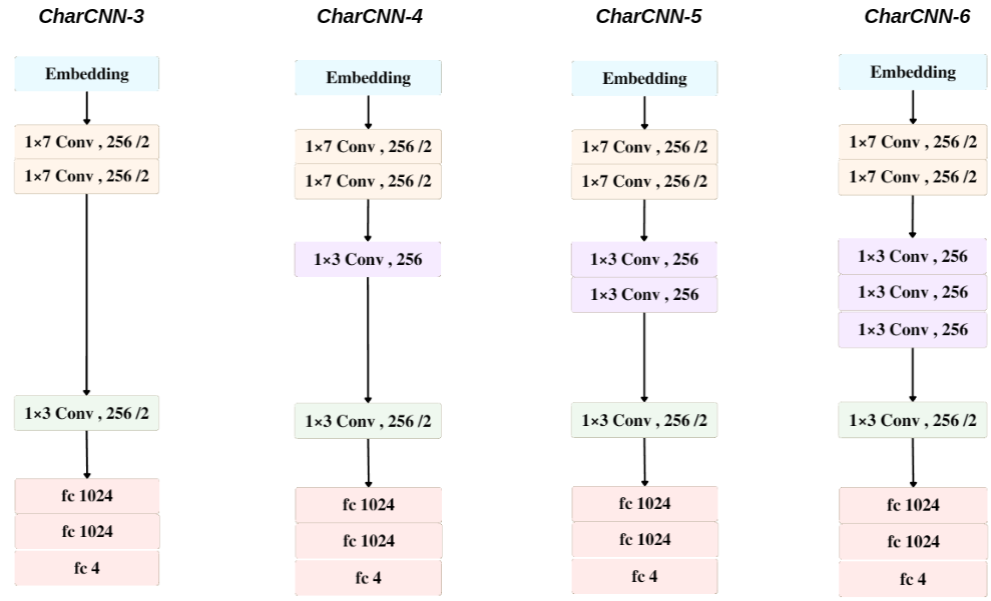
Note that the downsample layer in BasicBlock may not be needed sometimes. We use the four versions of ResNet-family : ResNet-20, ResNet-32, ResNet-44, ResNet-56, and the architectures of the nn are shown below :



The layer's parameters of ResNet-family include : [3,3,3], [5,5,5], [7,7,7], [9,9,9].

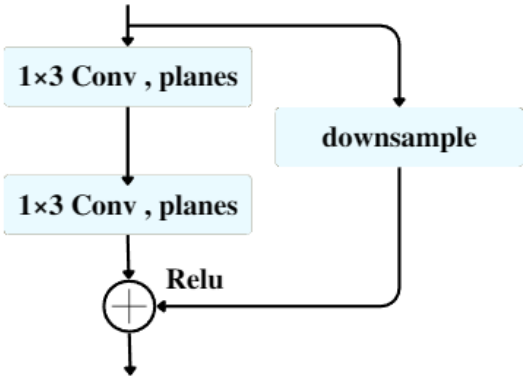
- CharCNN : Character-level convolutional neural networks proves that convolutional neural networks can also implement text classification with finer granularity.

  We use the four versions of CharCNN-family : CharCNN-3, CharCNN-4, CharCNN-5, CharCNN-6, and the architectures of the nn are shown below :

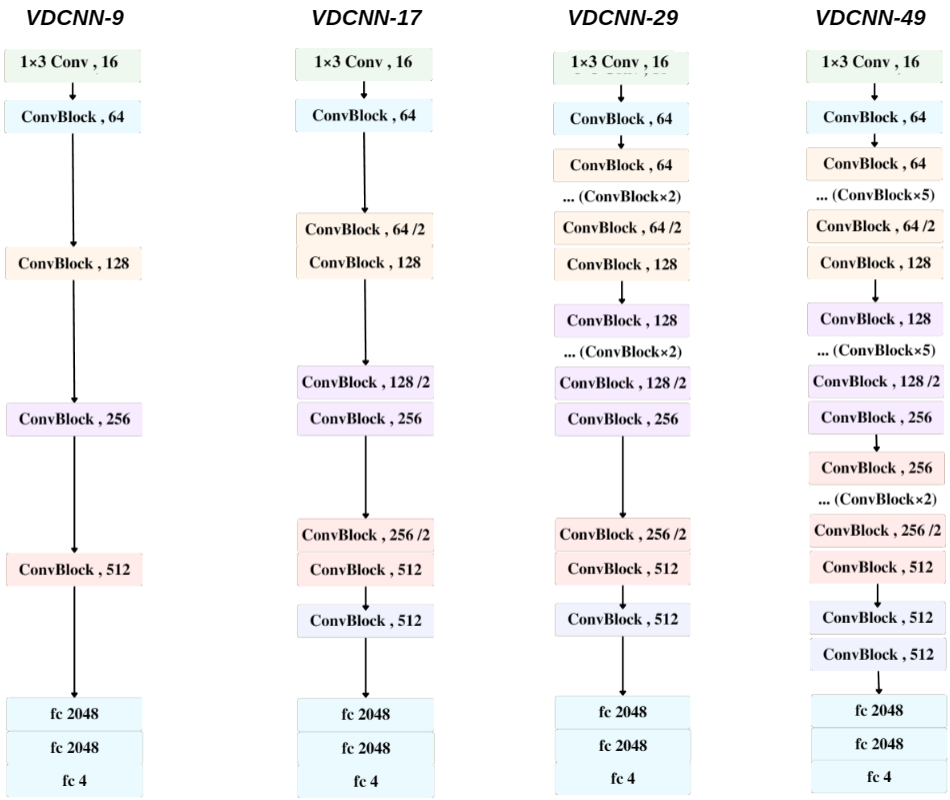


  In my mind, the CharCNN models designed in this project are easy to converge to local minimun on the AG_News dataset. The convergence process will be unexpected sometimes.

- VDCNN : Very Deep Convolutional neural network is similar to ResNet, allowing deeper networks to bring higher accuracy by computing residuals. The nn versions we use are stacks of ConvBlock, whose architecture is shown below (ConvBlock(planes)) :



Note that the downsample layer in ConvBlock may not be needed sometimes. We use the four versions of VDCNN-family : VDCNN-9, VDCNN-17, VDCNN-29, VDCNN-49, and the architectures of the nn are shown below :



The layer's parameters of VDCNN-family include : [0,0,0,0], [1,1,1,1], [4,4,1,1], [7,7,4,2]. According to my experimental results, the convergence process of VDCNN is oscillating, and the accuracy rate is not rising steadily.

Parameter Setting

- Optimizer : torch.optim.SGD
- Learning Rate : 0.01
- Momentum : 0.9
- Weight_decay : 5e-4
- num_clients : 8
- communication_round : 70

The optimizer for local training is SGD and its parameters (lr, momentum, weight_decay) are shown above. We set the number of clients in the FL system to 8, and each two clients share a version of the model-family (e.g. [0,1] - VGG-11, [2,3] - VGG-13, [4,5] - VGG-16, [6,7] - VGG-19) . Every client has a uid to identify them uniquely and the parameter server will use the uid to access these clients in order to aggregate the model of these clients.

Training Tips

Here are some important tips to help you implement the global training correctly :

- We will divide the dataset equally according to the number of clients in order to simulate that the data from different clients are personalized i.e. data heterogeneity.
- We will divide each client's dataset into 10 batches and select the batch in trun for local training in order to simulate that the data for each local training is not necessarily the same.
- To promise the data diversity of a batch for local training, the shuffle operation before dividing the dataset is necessary, e.g.

  The label of a batch before shuffle :

  ```
  tensor([7, 0, 7, 0, 7, 0, 7, 7, 0, 7, 0, 7, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
  7,
          0, 0, 0, 0, 7, 0, 7, 7, 7, 7, 7, 7, 0, 0, 7, 0, 7, 7, 0, 0, 7, 0, 0,
  0,
          0, 0, 7, 7, 7, 0, 7, 0, 7, 0, 0, 7, 0, 7, 0, 7])
  ```

  The label of a batch after shuffle :

  ```
  tensor([6, 4, 7, 7, 7, 6, 2, 0, 7, 0, 5, 9, 4, 0, 3, 7, 6, 3, 9, 0, 5, 7, 7,
  9,
          5, 5, 3, 3, 0, 6, 1, 7, 9, 5, 3, 7, 0, 7, 1, 0, 4, 6, 6, 3, 2, 9, 4,
  4,
          0, 2, 4, 8, 3, 3, 9, 1, 3, 1, 6, 2, 4, 8, 5, 7])
  ```

- Because the purpose of this project is mainly to prove the effectiveness of FlexiFed aggregation strategy, I do not spend too much attention on model design and data processing. Readers can further improve the accuracy of the involved model through some Data Augmentation operations.

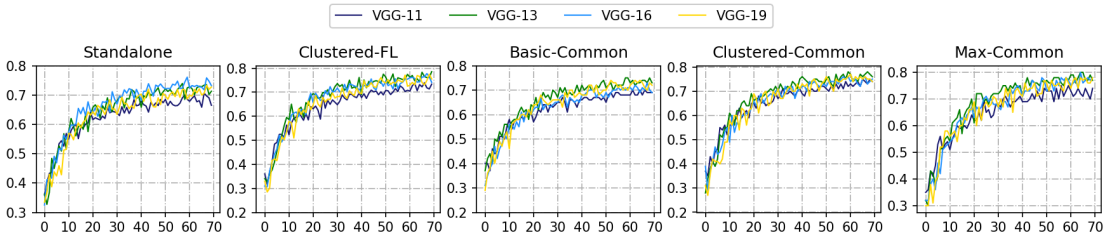
Results

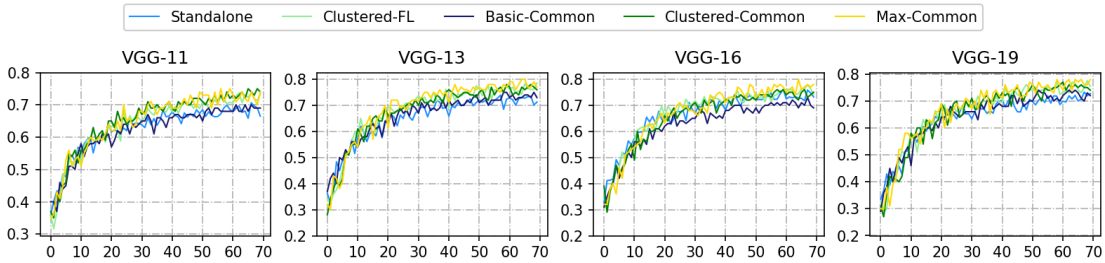
I will show you the results of federated learning for models with architecture heterogeneity on different datasets. Each result includes three pictures (the convergence process under different strategy and different model, the final accuracy) and one table (the final accuracy of the model trained by the FL system).

We assume that the number of clients in FL system is n, the accuracy of version v model is Acc_v, then there is :

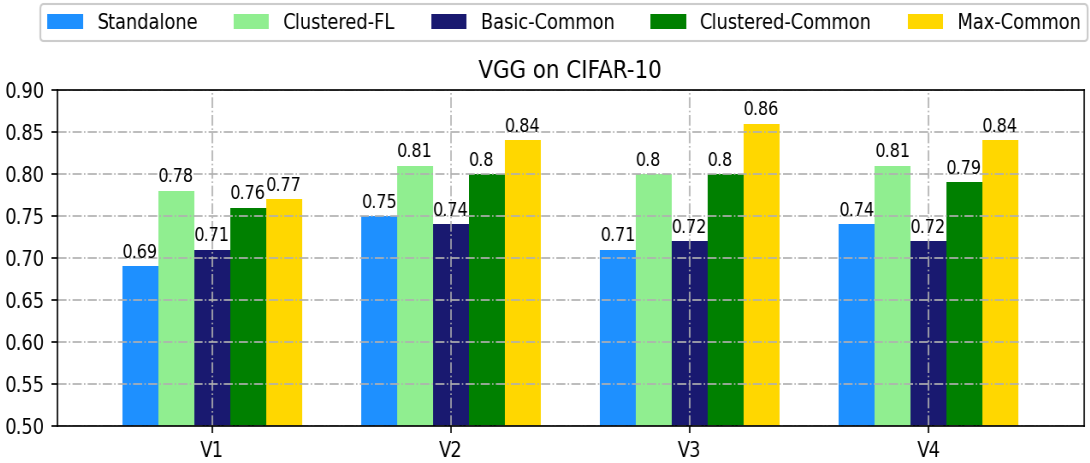
$$Acc_v = \frac{4}{n}\sum_{i=1}^{n/4} Acc_{v-i}$$

- VGG-family on CIFAR-10 :

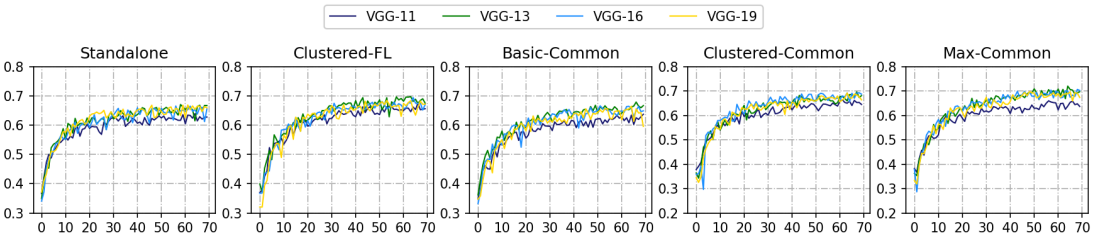


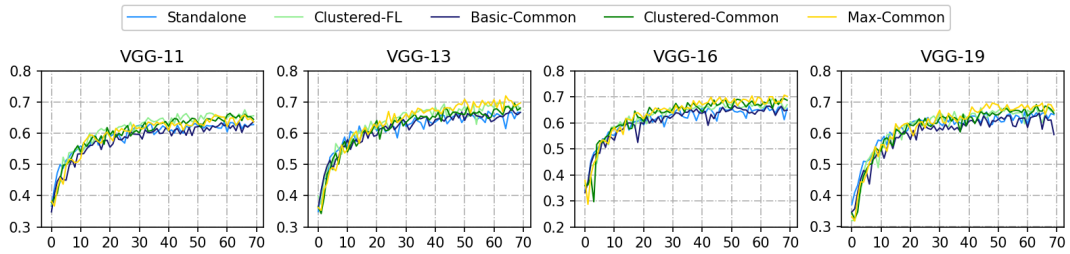


Scheme	V1	V2	V3	V4
Standalone	0.6936	0.7480	0.7084	0.7364
Clustered-FL	0.7848	0.8068	0.7960	0.8152
Basic-Common	0.7136	0.7368	0.7184	0.7152
Clustered-Common	0.7560	0.8000	0.7968	0.7892
Max-Common	0.7680	0.8428	0.8640	0.8428

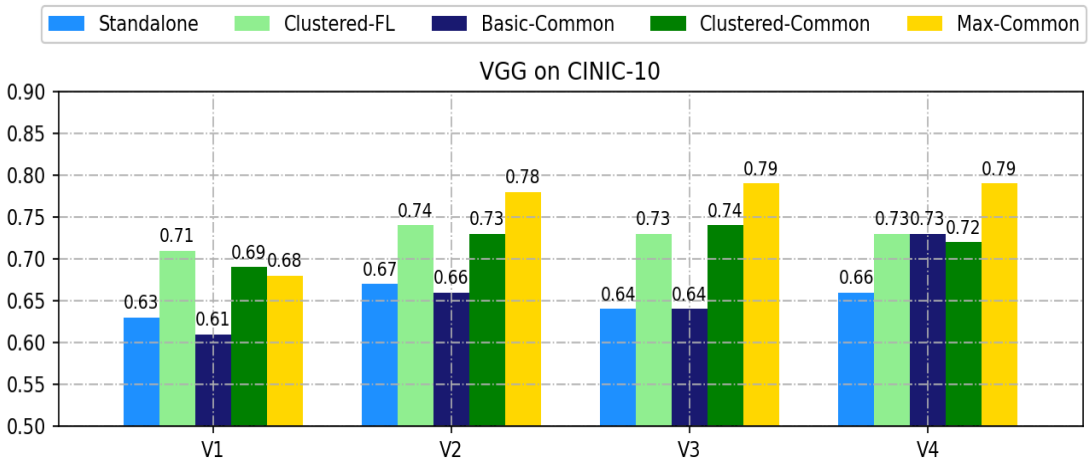


- VGG-family on CINIC-10 :

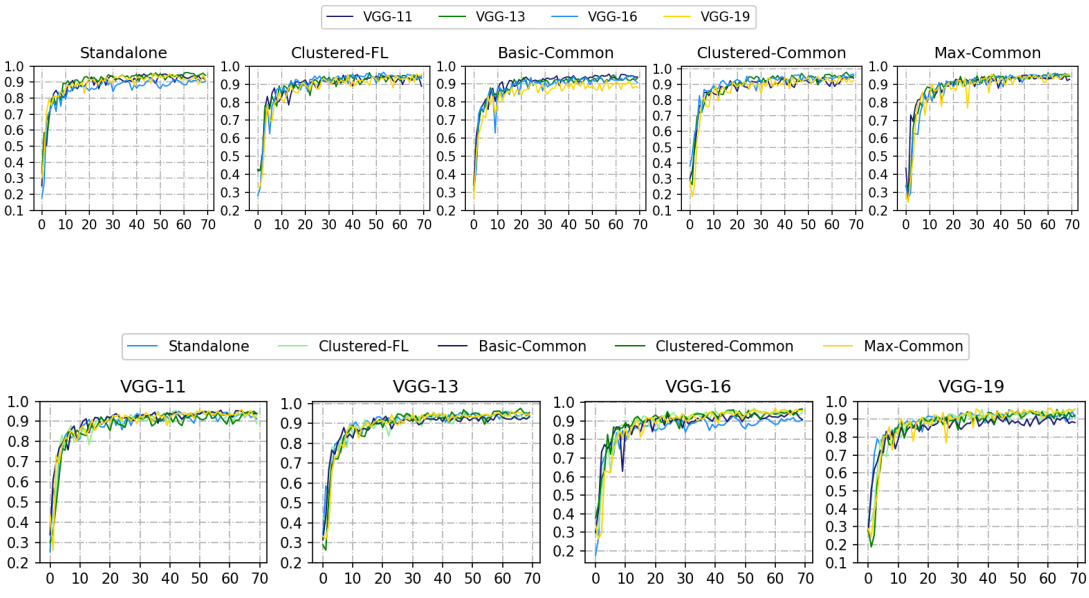




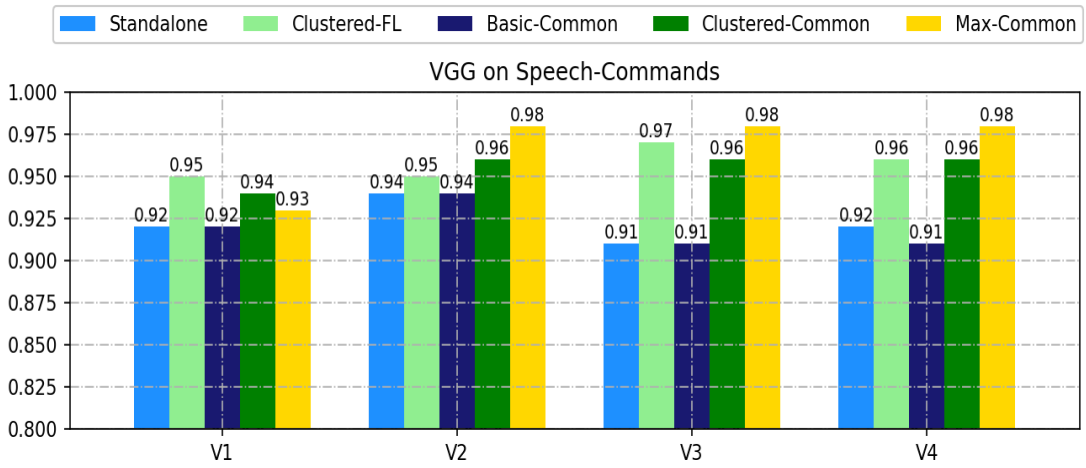
Scheme	V1	V2	V3	V4
Standalone	0.6319	0.6730	0.6434	0.6561
Clustered-FL	0.7111	0.7404	0.7292	0.7331
Basic-Common	0.6129	0.6619	0.6379	0.6225
Clustered-Common	0.6895	0.7317	0.7420	0.7228
Max-Common	0.6772	0.7772	0.7892	0.7874



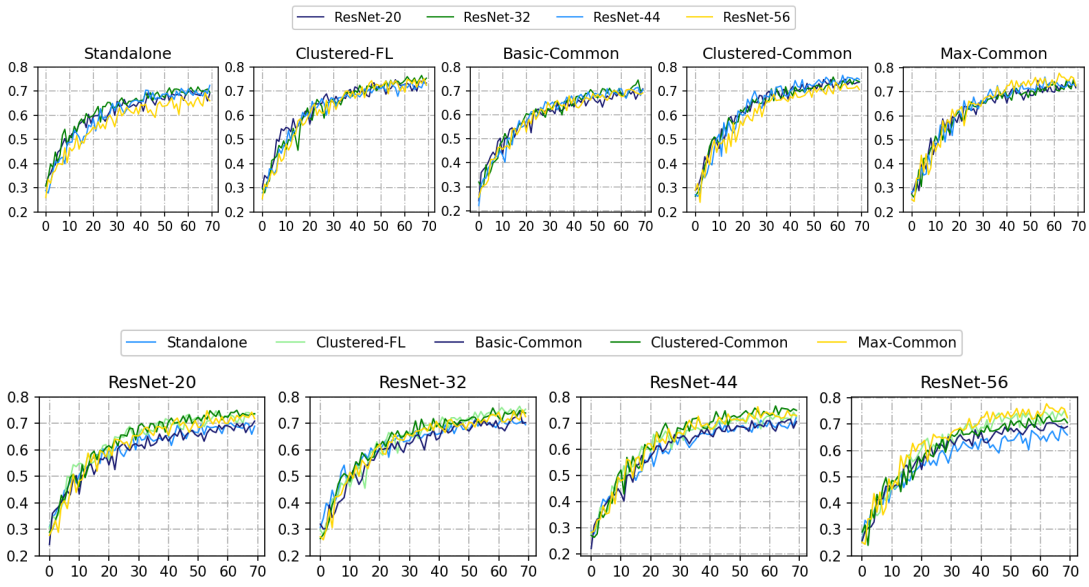
- VGG-family on Speech Commands :



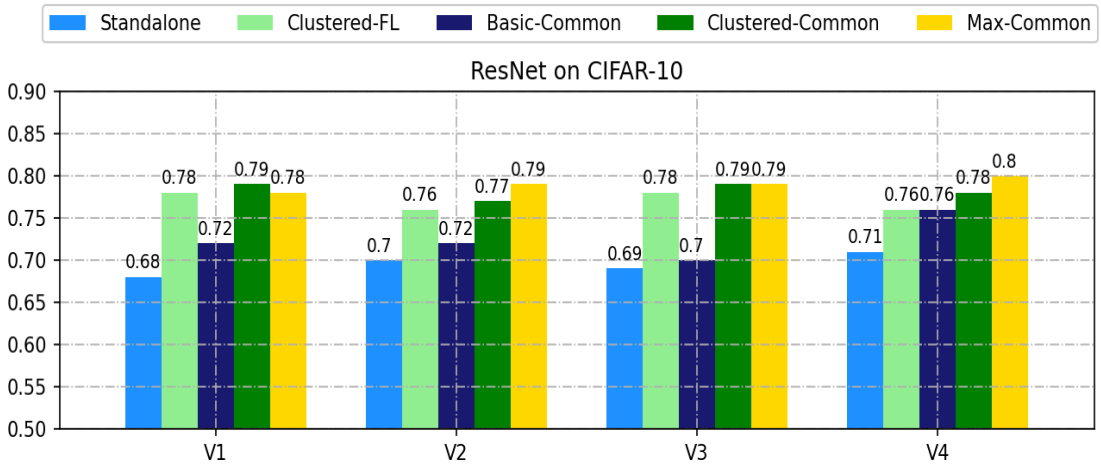
Scheme	V1	V2	V3	V4
Standalone	0.9207	0.9385	0.9077	0.9184
Clustered-FL	0.9516	0.9468	0.9681	0.9575
Basic-Common	0.9219	0.9385	0.9055	0.9126
Clustered-Common	0.9373	0.9563	0.9587	0.9598
Max-Common	0.9327	0.9787	0.9764	0.9764



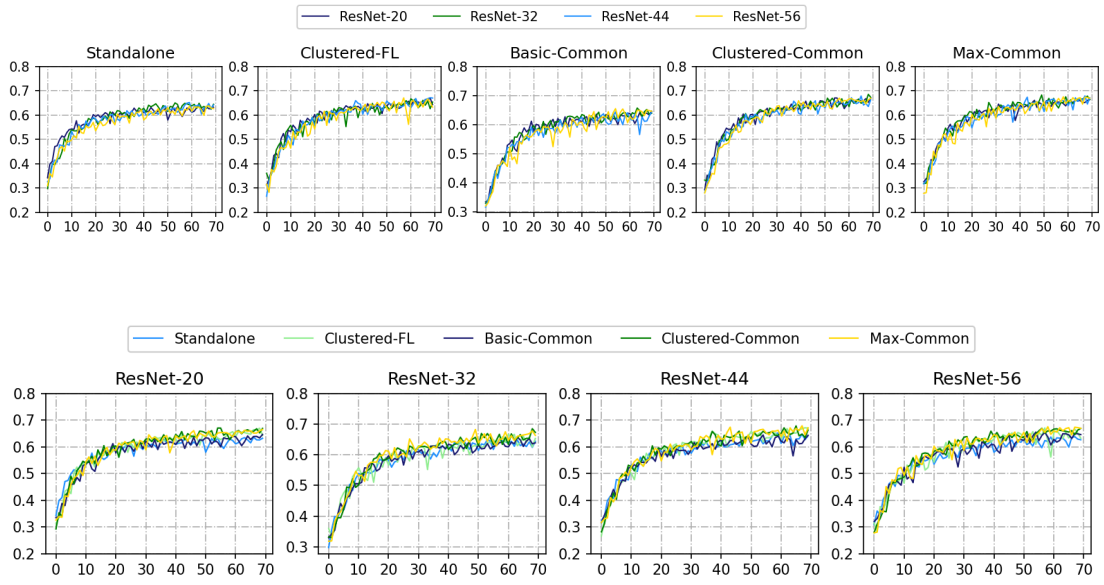
- ResNet-family on CIFAR-10 :



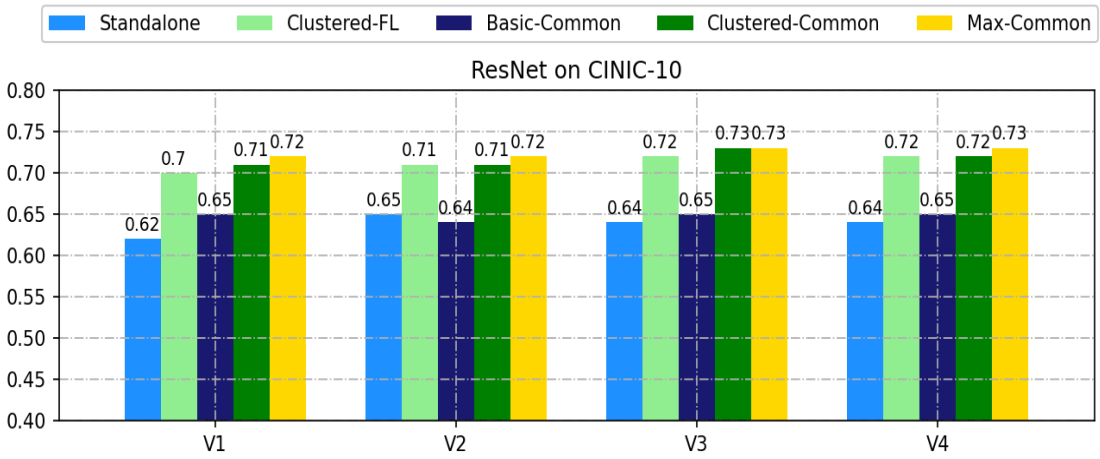
Scheme	V1	V2	V3	V4
Standalone	0.6764	0.7044	0.6884	0.7052
Clustered-FL	0.7824	0.7604	0.7796	0.7624
Basic-Common	0.7156	0.7164	0.7000	0.6908
Clustered-Common	0.7856	0.7672	0.7888	0.7752
Max-Common	0.7828	0.7944	0.7944	0.7956



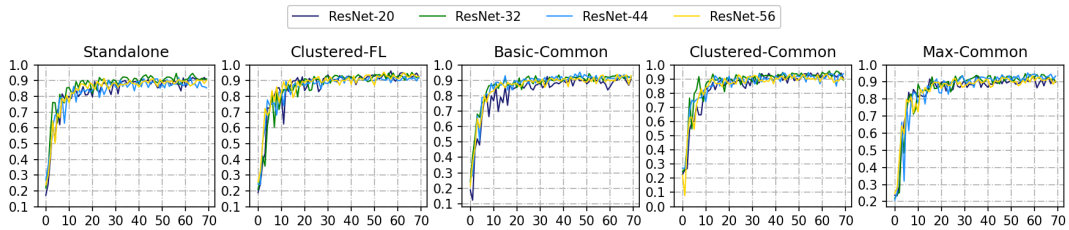
- ResNet-family on CINIC-10 :

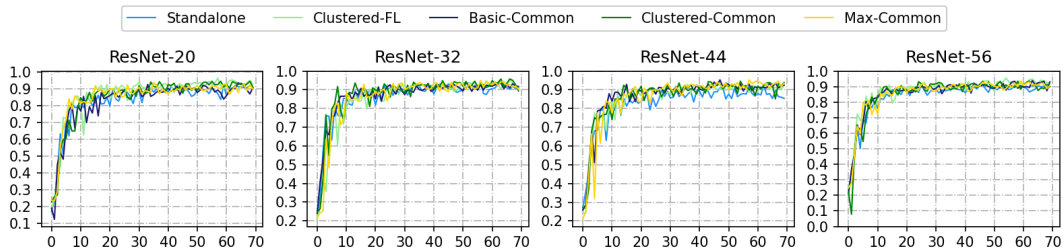


Scheme	V1	V2	V3	V4
Standalone	0.6190	0.6470	0.6402	0.6360
Clustered-FL	0.7014	0.7067	0.7156	0.7162
Basic-Common	0.6525	0.6385	0.6528	0.6539
Clustered-Common	0.7066	0.7119	0.7300	0.7249
Max-Common	0.7233	0.7252	0.7283	0.7281

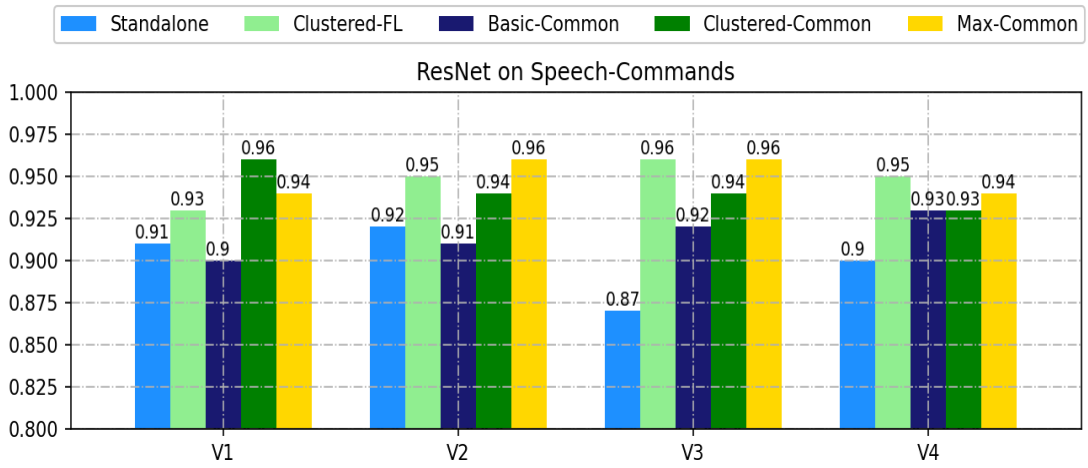


- ResNet-family on Speech Commands :

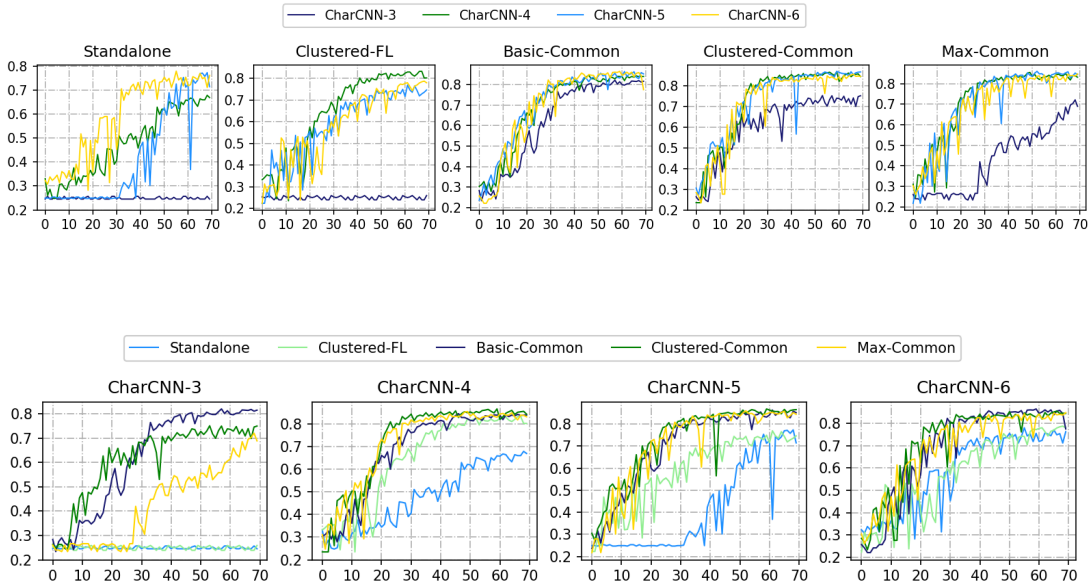


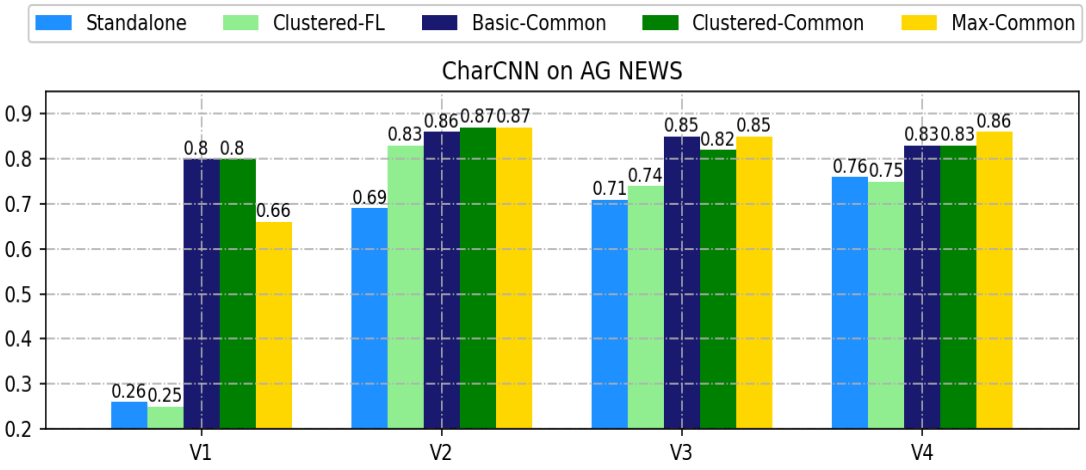


Scheme	V1	V2	V3	V4
Standalone	0.9101	0.9160	0.8723	0.9031
Clustered-FL	0.9326	0.9480	0.9350	0.9480
Basic-Common	0.9031	0.9054	0.9243	0.9291
Clustered-Common	0.9551	0.9445	0.94325	0.9303
Max-Common	0.9374	0.9563	0.9610	0.9433



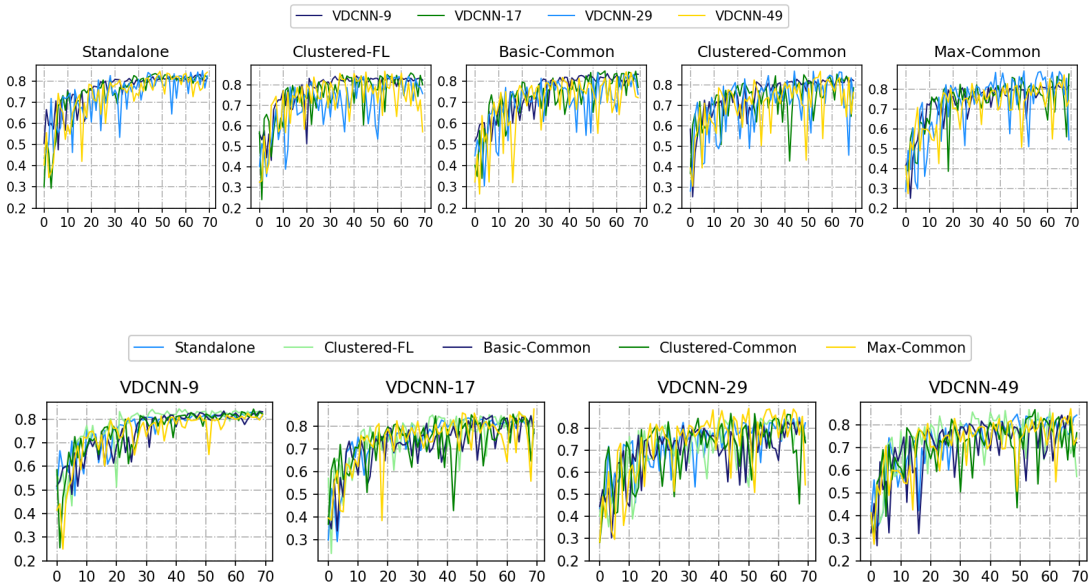
- CharCNN-family on AG News :



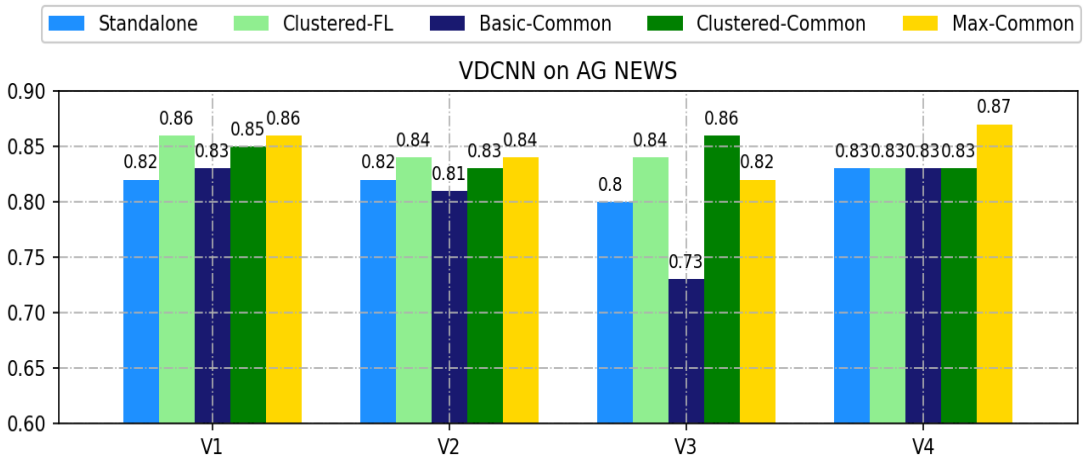


Scheme	V1	V2	V3	V4
Standalone	0.2589	0.6937	0.7095	0.7589
Clustered-FL	0.2500	0.8253	0.7400	0.7490
Basic-Common	0.7990	0.8579	0.8532	0.8326
Clustered-Common	0.8021	0.8711	0.8232	0.8342
Max-Common	0.6568	0.8705	0.8547	0.8590

- VDCNN-family on AG News :



Scheme	V1	V2	V3	V4
Standalone	0.8215	0.8205	0.7968	0.8279
Clustered-FL	0.8584	0.8384	0.8395	0.8332
Basic-Common	0.8305	0.8063	0.7284	0.8347
Clustered-Common	0.8500	0.8258	0.8621	0.8269
Max-Common	0.8558	0.8416	0.8216	0.8679



Comparation

I will compare my results with the result in Table1 of the paper and analyze the similarities and differences :

Dataset	Models	Scheme	Versions V1	V2	V3	V4
CIFAR-10	VGG	Standalone	0.69	0.75	0.71	0.74
		Clustered-FL	0.78	0.81	0.80	0.81
		Basic-Common	0.71	0.74	0.72	0.72
		Clustered-Common	0.76	0.80	0.80	0.79
		Max-Common	0.77	0.84	0.86	0.84
	ResNet	Standalone	0.68	0.70	0.69	0.71
		Clustered-FL	0.78	0.76	0.78	0.76
		Basic-Common	0.72	0.72	0.70	0.69
		Clustered-Common	0.79	0.77	0.79	0.78
		Max-Common	0.78	0.79	0.79	0.80
CINIC-10	VGG	Standalone	0.63	0.67	0.64	0.66
		Clustered-FL	0.71	0.74	0.73	0.73
		Basic-Common	0.61	0.66	0.64	0.73
		Clustered-Common	0.69	0.73	0.74	0.72
		Max-Common	0.68	0.78	0.79	0.79
	ResNet	Standalone	0.62	0.65	0.64	0.64
		Clustered-FL	0.70	0.71	0.72	0.72
		Basic-Common	0.65	0.64	0.65	0.65
		Clustered-Common	0.71	0.71	0.73	0.72
		Max-Common	0.72	0.73	0.73	0.73
AG NEWS	CharCNN	Standalone	0.26	0.69	0.71	0.76
		Clustered-FL	0.25	0.83	0.74	0.75
		Basic-Common	0.80	0.86	0.85	0.83
		Clustered-Common	0.80	0.87	0.82	0.83
		Max-Common	0.66	0.87	0.85	0.86
	VDCNN	Standalone	0.82	0.82	0.80	0.83
		Clustered-FL	0.86	0.84	0.84	0.83
		Basic-Common	0.83	0.81	0.73	0.83
		Clustered-Common	0.85	0.83	0.86	0.83
		Max-Common	0.86	0.84	0.82	0.87
Speech Commands	VGG	Standalone	0.92	0.94	0.91	0.92
		Clustered-FL	0.95	0.95	0.97	0.96
		Basic-Common	0.92	0.94	0.91	0.91
		Clustered-Common	0.94	0.96	0.96	0.96
		Max-Common	0.93	0.98	0.98	0.98
	ResNet	Standalone	0.91	0.92	0.87	0.90
		Clustered-FL	0.93	0.95	0.96	0.95
		Basic-Common	0.90	0.91	0.92	0.93
		Clustered-Common	0.96	0.94	0.94	0.93
		Max-Common	0.94	0.96	0.96	0.94

Dataset	Models	Scheme	Versions V1	V2	V3	V4
CIFAR-10 [24]	VGG [46]	Standalone	64.4	64.8	65.2	65.6
		Clustered-FL	78.2	80.4	80.8	81.2
		Basic-Common	65.2	66.8	67.2	68.2
		Clustered-Common	80.2	82.4	83.2	83.6
		Max-Common	80.6	85.2	86.6	86.8
	ResNet [14]	Standalone	57.2	57.8	58.8	59.2
		Clustered-FL	73.6	74.6	75.2	75.8
		Basic-Common	66.4	67.6	67.8	68.4
		Clustered-Common	78.4	79.2	80.6	80.8
		Max-Common	78.8	79.6	83.4	84.2
CINIC-10 [8]	VGG	Standalone	44.4	45.6	47.4	49.2
		Clustered-FL	58.8	59.8	60.4	60.6
		Basic-Common	48.8	50.8	51.2	51.4
		Clustered-Common	60.8	62.8	63.4	63.8
		Max-Common	61.4	67.6	67.8	68.2
	ResNet	Standalone	46.2	47.2	47.8	49.4
		Clustered-FL	58.2	58.6	59.8	60.4
		Basic-Common	49.8	51.4	51.8	52.2
		Clustered-Common	60.8	61.4	63.4	64.2
		Max-Common	61.6	64.2	66.2	66.6
AG NEWS [65]	CharCNN [65]	Standalone	51.9	53.2	65.1	70.2
		Clustered-FL	76.2	77.7	84.1	84.7
		Basic-Common	84.6	85.6	85.7	86.1
		Clustered-Common	85.2	85.6	86.1	86.3
		Max-Common	85.9	86.5	86.7	87.6
	VDCNN [6]	Standalone	73.2	75.8	77.8	78.6
		Clustered-FL	84.6	85.2	86.2	86.4
		Basic-Common	78.2	79.6	80.4	80.6
		Clustered-Common	84.8	86.6	87.8	88.2
		Max-Common	88.2	89.2	89.6	90.2
Speech Commands [54]	VGG	Standalone	77.5	78.2	80.5	81.5
		Clustered-FL	80.2	81.8	83.4	85.8
		Basic-Common	78.6	80.4	81.2	82.4
		Clustered-Common	82.4	84.6	86.2	86.4
		Max-Common	82.6	86.6	87.4	89.8
	ResNet	Standalone	75.2	78.6	79.4	82.8
		Clustered-FL	79.2	82.0	83.6	84.2
		Basic-Common	83.6	87.6	88.2	89.2
		Clustered-Common	84.8	88.8	89.2	89.8
		Max-Common	85.2	89.6	90.8	91.4

- Similarities :

  Although the differences between the various aggregation strategies are not obvious in the convergence process figures, the value relationship for accuracy is almost the same as that in Table1, which satisfies :

$$Basic - Common < Clustered - Common < Max - Common$$

$$Standalone \approx Basic - Common$$

$$Clustered - FL \approx Clustered - Common$$

This demonstrates the usefulness of FlexiFed. The idea "aggragate the model in FL system to the maximum extent" does work and the gains from Max-Common strategy are significant.

- Differences :
  - The accuracy of ResNet family on CIFAR-10 is slightly lower than Table1's :

    I think this is due to the lack of communication rounds. When training to 70 rounds, the accuracy is just stable, we can see that there is still a small upward trend. ResNet family needs more communication rounds than VGG family to get higher accuracy.
  - The accuracy of models on Speech Commands under the Standalone, Clustered-FL, Basic-Common strategy is much higher than Table1's :

    I think this is due to the data augmentation before local training. I use the STFT to transform the 1d audio to 2d matrix, so that we don't have to adjust the model and this operation contribute much to the higher accuracy.
  - The accuracy of CharCNN family on AG NEWS is slightly lower than Table1's :

    CharCNN can easily fall into the local minima (25%, 50%, 75%) during the convergence process especially the CharCNN-3 under the Standalone strategy. I think this is due to the presence of xor-like operations during training, which will bring the difficulty for models with simple architechiture to converge normally. But I found that the aggregation strategy of FlexiFed will help the model to escape the local minimum when there is a well convergent model in the FL system, which confirms the usefulness of federated learning and FlexiFed again.
  - The convergence process of VDCNN is oscillating :

    I think that's inherent in the model, because the convergence process of the model is oscillating even under the standalone strategy. Although the higher the degree of aggregation of the model, the more obvious the oscillation, it does not affect the final high accuracy of the model.

References

Here are some repositories and articles that I refer to during the implementation of the project :

- Project Structure :

  fio1982/FlexiFed (github.com)
- The architecture of ResNet :

  pytorch_resnet_cifar10: Proper implementation of ResNet-s for CIFAR10/100 in pytorch that matches description of the original paper.
- Loading of Speech Commands dataset :

  pytorch-speech-commands: Speech commands recognition with PyTorch | Kaggle 10th place solution in TensorFlow Speech Recognition Challenge
- The architecture of CharCNN :

charcnn-classification: Character-level Convolutional Networks for Text Classification in Pytorch

- The architecture of VDCNN :

  Very-deep-cnn-pytorch: Very deep CNN for text classification

- Loading of AG News dataset :

  pytorch-char-cnn-text-classification: A pytorch implementation of the paper "Character-level Convolutional Networks for Text Classification"