Report of Machine Learning Project

Zhixin Ling, 516021910495 Yifeng Gao,0000000000000

I. Main Ideas

In this project we use deep methods to perform classification work on Fer2013 dataset. We classify people's face images into seven categories: Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral. We try different network architectures and select network parameters that result in the best accuracy on the given public test set. Finally, we test our networks and parameters on the given private test set to investigate the performance. Also, we try cropping the face segment of the images first to eliminate some unnecessary noises to see whether the classification can improve.

We find networks of ResNet18 and VGG19 achieve the best performance on the given private test dataset with the selected parameters without face cropping. The best resulted accuracy reaches 0.73, reaching the state-of-art achievement. With face cropping, the overall resulted accuracies reduce by about 0.03. However, on several classes, the resulted classification accuracies improve greater than 0.02.

II. Methods

A. LeNet

LeNet is a classical neural network. It is the first complete proposed neural network architecture, including convolution layers, pooling layers, fully connected layers and neuron activation mechanism. A typical version of LeNet is illustrated in 1. The network is simple but can be very effective for some requirements. LeNet's training needs far less computation compared with some later complicate networks. So, LeNet can always worth a try in our classification work.

Compared with the first proposed version of LeNet, we use RELU instead of sigmoid function for neuron activation and also apply batch normalization to ensure a fast and stable convergence.

B. VGG

VGG, short for Visual Geometry Group Network, is far deeper than LeNet. More convolution kernels are applied; methods of neuron dropout and batch normalization are introduced; larger fully connected layers are used. A typical version of VGG(VGG16) is illustrated in 2. We try several architectures of VGG of different depths to explore whether a deeper network brings a better classification performance.

C. ResNet

ResNet, short for residual neural network, won the first prize in ILSRVC2015. The special feature of ResNet is that it creatively proposed shortcut connections that allow a former layer to have additional connections, bypassing adjacent layers, leading to the layers behind, as illustrated in 3. The idea greatly ease decreasing of accuracies caused by increasing of network depths, thus allowing the network as deep as possible. As a result, ResNet have far more convolution layers than VGG and also bigger convolution kernels.

D. Face Detection And Cropping

The samples of our dataset are not all well collected. Faces in some images may be misplaced in a corner and the main contents of the image can be something uncorrelated to the classification work. So we try to detect the area in a image where the face exists and crop the area as a sample. The old image is discarded, of course. For the work of face detection, we use traditional methods. We use a trained classifier cascaded from strong subclassifiers trained with extracted Haar features of images using Adaboost algorithms.

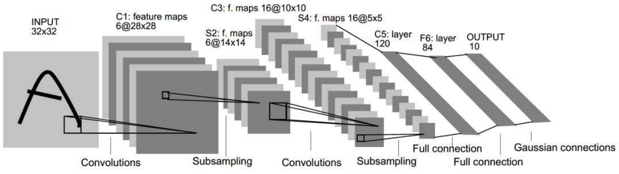

Fig. 1: Typical Lenet Architecture

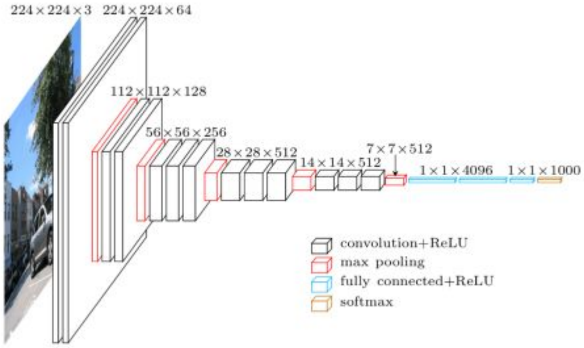

Fig. 2: VGG16 Architecture

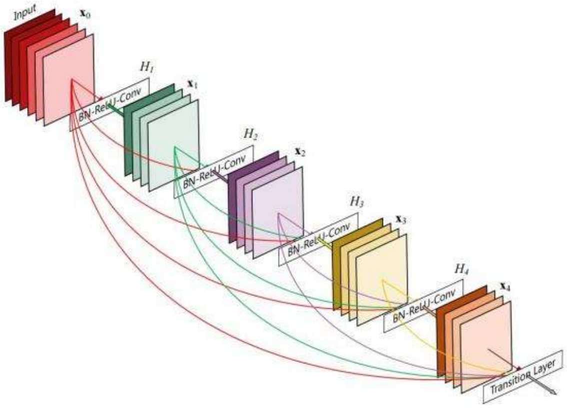

Fig. 3: Residual Connections

E. Data Augmentation

Data augmentation is used to enlarge the training dataset, creating more training samples, to make the trained model more robust and ease overfitting. Symmetry and random cropping are used for data augmentation. That is, for a given training sample image, we flip the image randomly and crop it to a given size. For a test sample, we do not flip but only crop it.

III. Algorithms

A. Preprocessing

Before training, we need to preprocess our data. This includes face detection and cropping, if required, and data augmentation. The algorithm is elaborated in 1.

B. Network Training

Because of hardware limitations, we train our model in batches. And learning rate annealing is also applied to stabilize the convergence. After each epoch, we estimate the classification accuracy on the public test set and decide whether or not to save the current network parameters. The algorithm is elaborated in 2.

We use cross entropy as loss for back propagation. Compared to mean squared error, cross entropy suffers less from gradient vanishing and help greatly update of network parameters, especially for multi-class classification requirements.

IV. Experimental Settings

A. Convention of Notations

Before going on, we present our common notations to avoid confusion in table I. The middle column Fixed Value indicates we have a fixed value for the parameter and we use $-$ to indicate the value is not fixed.

Algorithm 1: Preprocess

```
Input:
    An $b \times m$ matrix $B$, a batch of raw data of size $b$.
    $F_{DC}$, boolean, whether or not to detect and crop the face part of the image.
    $l'$, side length of output image.
Output:
    A $b \times l' \times l'$ matrix $X$
for Every row $R$ in $B$ do
    /* Reshape $R$ to be an image. */
    Reshape $R$ to be an $l \times l$ matrix, where $l = \sqrt{m}$;
    if $F_{DC}$ is true then
        Detect the area where the face exists in $R$;
        Discard the rest area of $R$;
        Resize $R$ to be $l \times l$;
    /* Perform data augmentation. */
    Randomly select a $l' \times l'$ area in $R$;
    $R \leftarrow$ the selected part;
    Randomly decide whether or not to flip the image horizontally with a chance of 0.5;
$X \leftarrow B$;
return $X$;
```

Algorithm 2: Network Training

```
Input:
    $(X^{tr}, Y^{tr})$, the training set.
    $(X^{te}_{pub}, Y^{te}_{pub})$, the public test set.
    $F_{DC}$, boolean, whether or not to detect and crop the face part of the image.
    $\theta$, parameters of the given network.
    $Ep_m$, the maximum epoch.
    $l'$, side length of image for training.
Output:
    $\theta*$, trained parameters of the given network.
Initialize the $\theta$ and $\theta^* \leftarrow \theta$;
Initialize learning rate $l_r$;
$acc^{te*}_{pub} \leftarrow 0.0$;
for $e = 1 \rightarrow Ep_m$ do
    for Each batch $i$ in $(X^{tr}, Y^{tr})$ do
        Select a batch $(X^{tr}_i, Y^{tr}_i)$ from $(X^{tr}, Y^{tr})$;
        $Preprocess(X^{tr}_i, F_{DC}, l')$;
        Calculate array $q$ with $q_j \leftarrow f(\theta, X^{tr}_{i,j})$, where $j = 1, 2, \cdots, b$ and $f$ denotes the function corresponding to the given network;
        $q \leftarrow softmax(q)$;
        $loss \leftarrow -\frac{1}{b} \sum_{j=1}^{b} Y^{tr}_{i,j} \log q_j$;
        $\theta \leftarrow \theta - l_r \times \frac{\partial loss}{\partial \theta}$;
    Calculate $q$ for all batches of $(X^{te}_{pub}, Y^{te}_{pub})$;
    $acc^{te}_{pub} \leftarrow \frac{1}{s^{te}_{pub}} \sum_j (argmax(q_j) == Y^{te}_{pub,j})$, where $s^{te}_{pub}$ is total size of public training set;
    if $acc^{te}_{pub} \geq acc^{te*}_{pub}$ then
        $acc^{te*}_{pub} \leftarrow acc^{te}_{pub}$;
        $\theta^* \leftarrow \theta$;
    Anneal $l_r$;
return $\theta^*$;
```

TABLE I: Convention of Notations

Notation	Fixed Value	Description
l	48	Side length of raw input images.
l'	44	Cropped side length of input images.
F_{DC}	-	Whether or not to detect and crop the face part of the image.
D_o	0.5	Neurons' drop out rate when training(if a neuron is allowed so).
Ep_m	250	Maximum number of epochs.
$l_r(e)$	$0.01 \times 0.9^{e/5}$	Learning rate in a given epoch e.
b	-	Batch size.
$S_{tr} = (X^{tr}, Y^{tr})$	✓	Training set.
$S_{pub} = (X^{te}_{pub}, Y^{te}_{pub})$	✓	Public test set.
$S_{pri} = (X^{te}_{pri}, Y^{te}_{pri})$	✓	Private test set.
e^*	-	The epoch when the best accuracy on public test set is last achieved.
$acc^{tr}(e)$	-	Training accuracy in a given epoch e.
$acc^{te}_{pub}(e)$	-	Private test accuracy in a given epoch e.
$acc^{te}_{pri}(e)$	-	Public test accuracy in a given epoch e.

TABLE II: Structures of LeNet5 and VGG

LeNet5	VGG11	VGG13	VGG16	VGG19
input	input	input	input	input
conv5-6	conv3-64	conv3-64	conv3-64	conv3-64
maxpool	maxpool	conv3-64	conv3-64	conv3-64
conv5-16	conv3-128	maxpool	maxpool	maxpool
maxpool	maxpool	conv3-128	conv3-128	conv3-128
FC-144	conv3-256	conv3-128	conv3-128	conv3-128
FC-84	conv3-256	maxpool	maxpool	maxpool
FC-7	maxpool	conv3-256	conv3-256	conv3-256
soft-max	conv3-512	conv3-256	conv3-256	conv3-256
	conv3-512	maxpool	conv3-256	conv3-256
	maxpool	conv3-512	maxpool	conv3-256
	conv3-512	conv3-512	conv3-512	maxpool
	conv3-512	maxpool	conv3-512	conv3-512
	maxpool	conv3-512	conv3-512	conv3-512
	FC-512	conv3-512	maxpool	conv3-512
	FC-7	maxpool	conv3-512	conv3-512
	soft-max	FC-512	conv3-512	maxpool
		FC-7	conv3-512	conv3-512
		soft-max	maxpool	conv3-512
			FC-512	conv3-512
			FC-7	conv3-512
			soft-max	maxpool
				FC-512
				FC-7
				soft-max

B. *Detailed Network Structures*

We conduct eight experiments with different settings on six different networks. We use LetNet5, VGG11, VGG13, VGG16, VGG19 and ResNet18, as shown in table II and figure 4. In table II, we denote a convolution layer as ”conv¡kernel size¿-¡kernel numbers¿” and a fully connected layer as ”FC-¡neuron numbers¿”. RELU is used for neuron activation. Padding is applied with every convolution layers and batch normalization is used after convolution.

Compared with the LeNet5 structure used for mnist digits classification, we adjust the fully connected layers to adapt to our dataset and apply batch normalization.

Compared with standard VGG structures proposed by Karen Simonyan, et al., we, use only one fully connected layer(”FC-512”) after convolution layers except the output layer while Karen Simonyan, et al. use two 4096-neuron layers. We do not need such complicated fully connected layers because we have only seven classes while they have one thousand.

Since ResNet is more difficult to describe, we turn to tensorboard for visualization. The detailed parameters are elaborated in III. We denote a convolution layer using n s-sized kernels as (s, n) and a sequenced building blocks composed of k convolution layer as [(s_1, n_1),(s_2, n_2), $\cdots$, (s_k, n_k)] $\times a$. Within every building block, there is a residual connection from block input to block output, bypassing the contained convolution layers, as illustrated in the right block of figure 4.

For every fully connected layer, we drop out the neurons with a rate of $D_o = 0.5$.

C. *Hyper-Parameters of Experiments*

We conduct eight experiments regard of different networks, whether or not crop faces in preprocessing and the batch size used for training. The details are illustrated in table IV.

TABLE III: Structures of ResNet18

Layer Name	Blocks
conv1	(3,64)
conv2.x	$[(3,128),(3,128)]\times 2$
conv3.x	$[(3,256),(3,256)]\times 2$
conv4.x	$[(3,512),(3,512)]\times 2$
FC1	FC-512
output	soft-max

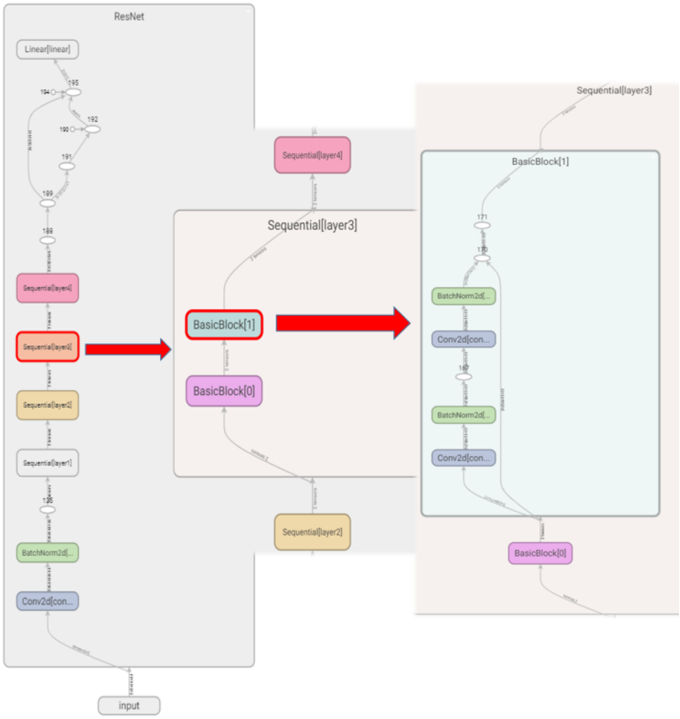

Fig. 4: Structures of ResNet18

V. Results

In this part, we'll first show an overview of our results in table V. Among the eight experiments, ExRes18 achieve $acc^{te}_{pri}(e^*) = 0.73$. $acc^{te}_{pri}(e^*)$ of all experiments except the two that use LeNet5 reach 0.70, the state-of-art result. Besides, in all experiments, there is a phenomena that $acc^{te}_{pub}(e^*) \leq acc^{te}_{pri}(e^*)$. The reason could be that joint distribution of S_{pri},$P(X^{te}_{pri}, Y^{te}_{pri})$, is more close to distribution of S_{tr},$P(X^{tr}, Y^{tr})$, than that of S_{pub},$P(X^{te}_{pub}, Y^{te}_{pub})$. And we can conclude from the low $acc^{tr}(e^*)$ from ExLenet5 and ExLenet5-DC that LeNet5 structure is too simple to classify the our face expression dataset. The underfitting is very obvious. By observing and comparing results of ExLenet5 and ExLenet5-DC, ExVGG19 and ExVGG19-DC, the face detection and cropping, to our disappointment, lowers the $acc^{te}_{pri}(e^*)$ by about 0.01. The face detection and cropping doesn't help. And, it is unlikely that increasing depth of VGG networks can necessarily bring improvement on the network's performance.

Now, we present private test accuracy confusion matrix of some experiments. For a confusion matrix M, each element $M_{i,j}$ denotes the ratio of samples belonging to class i classified into class j, where $i, j = 1, 2, 3, 4, 5, 6, 7$, representing expression of angry, disgust, fear, happy, sad, surprise and neutral respectively.

Comparing confusion matrices of ExVGG19 and ExVGG19-DC in 1 and 2, ExVGG19-DC is not so frustrating since classification accuracies of VGG19-DC on class 5 and class 6 overcome those of ExVGG19 by 0.05 and 0.02 respectively.

TABLE IV: Hyper-Parameters of Experiments

Experiment Name	ExLenet5	ExLenet5-DC	ExVGG11	ExVGG13	ExVGG16	ExVGG19	ExVGG19-DC	ExRes18
F_{DC}	×	✓	×	×	×	×	✓	×
b	64	64	64	64	64	64	64	32
Network	LeNet5	LeNet5	VGG11	VGG13	VGG16	VGG19	VGG19	ResNet18

TABLE V: An Overview of Experiment Results

Experiment Name	ExLenet5	ExLenet5-DC	ExVGG11	ExVGG13	ExVGG16	ExVGG19	ExVGG19-DC	ExRes18
e^*	249	250	204	196	193	249	250	170
$acc^{tr}(e^*)$	0.68	0.68	0.99	0.99	0.99	0.99	0.99	0.99
$acc^{te}_{pub}(e^*)$	0.63	0.62	0.70	0.71	0.71	0.71	0.69	0.71
$acc^{te}_{pri}(e^*)$	0.64	0.63	0.71	0.72	0.71	0.72	0.71	0.73

We think each experiment configuration has its own advantages. But all of them do best in class 6(surprise) and do worst in class 3(fear). we have opinions as follow:

1) Our model may not be complicated enough to recognize expression of fear. And fear is the most undistinguished among all expressions. Actually, even in the training samples of class 3, the classification error is also the highest.
2) The samples may be mislabeled. Even human can sometimes misunderstand face expressions.

Finally, we place the training histories of our experiments in 5. We have discoveries as follow:

1) By observing the figures, we find we do not always select the best e^* for the best $acc^{te}_{pri}(e^*)$. For example, there are several moments when $acc^{te}_{pri}(e^*)$ reach 0.73 during the training of ExVGG19 but our rule of selecting network parameters just failed to grab those moments. We have to admit that there are some error in our codes so that we failed to record the accuracies as precise as possible.
2) It is discovered that loss curves using LeNet5 are more unstable, especially when using face cropping for preprocessing.
3) Except the two experiments using LeNet5, test losses of all experiments first decrease and the increase while training losses are always decreasing. The three curves intersect at roughly one point. Before the point, training accuracies and test accuracies increase quickly and consistently, while, after the point, training accuracies still continue the old trend but test accuracies improve slowly.

$$M(ExVGG19) = \begin{pmatrix} 0.6191 & 0.0102 & 0.1141 & 0.0305 & 0.1161 & 0.0163 & 0.0937 \\ 0.1818 & 0.7273 & 0.0545 & 0.0182 & 0. & 0. & 0.0182 \\ 0.0909 & 0. & 0.5852 & 0.0303 & 0.1515 & 0.0606 & 0.0814 \\ 0.0148 & 0. & 0.0114 & 0.9101 & 0.0205 & 0.0159 & 0.0273 \\ 0.101 & 0. & 0.1279 & 0.0303 & 0.5707 & 0.0034 & 0.1667 \\ 0.0192 & 0.0024 & 0.0986 & 0.0312 & 0.0144 & 0.8101 & 0.0240 \\ 0.0447 & 0.0016 & 0.0495 & 0.0367 & 0.1262 & 0.008 & 0.7332 \end{pmatrix} \tag{1}$$

$$M(VGG19DC) = \begin{pmatrix} 0.6191 & 0.0061 & 0.112 & 0.0224 & 0.1365 & 0.0163 & 0.0876 \\ 0.1818 & 0.7091 & 0.0364 & 0.0364 & 0. & 0.0182 & 0.0182 \\ 0.0985 & 0. & 0.5587 & 0.0303 & 0.1496 & 0.0644 & 0.0985 \\ 0.0171 & 0. & 0.0159 & 0.8919 & 0.025 & 0.0171 & 0.0330 \\ 0.0774 & 0.0017 & 0.1077 & 0.0539 & 0.6229 & 0.0051 & 0.1313 \\ 0.0192 & 0. & 0.0769 & 0.0337 & 0.0168 & 0.8341 & 0.0192 \\ 0.0415 & 0.0016 & 0.0511 & 0.0415 & 0.1534 & 0.0112 & 0.6997 \end{pmatrix} \tag{2}$$

$$M(Res18) = \begin{pmatrix} 0.6558 & 0.0041 & 0.1018 & 0.0204 & 0.1161 & 0.0122 & 0.0896 \\ 0.1636 & 0.7273 & 0.0364 & 0.0364 & 0. & 0.0182 & 0.0182 \\ 0.0966 & 0.0038 & 0.5795 & 0.0246 & 0.1383 & 0.0606 & 0.0966 \\ 0.0148 & 0.0011 & 0.0171 & 0.8942 & 0.0171 & 0.0148 & 0.041 \\ 0.0741 & 0. & 0.1229 & 0.0269 & 0.6128 & 0.0034 & 0.1599 \\ 0.0144 & 0. & 0.0817 & 0.0312 & 0.024 & 0.8317 & 0.0168 \\ 0.0351 & 0.0016 & 0.0495 & 0.0415 & 0.1182 & 0.0096 & 0.7444 \end{pmatrix} \tag{3}$$

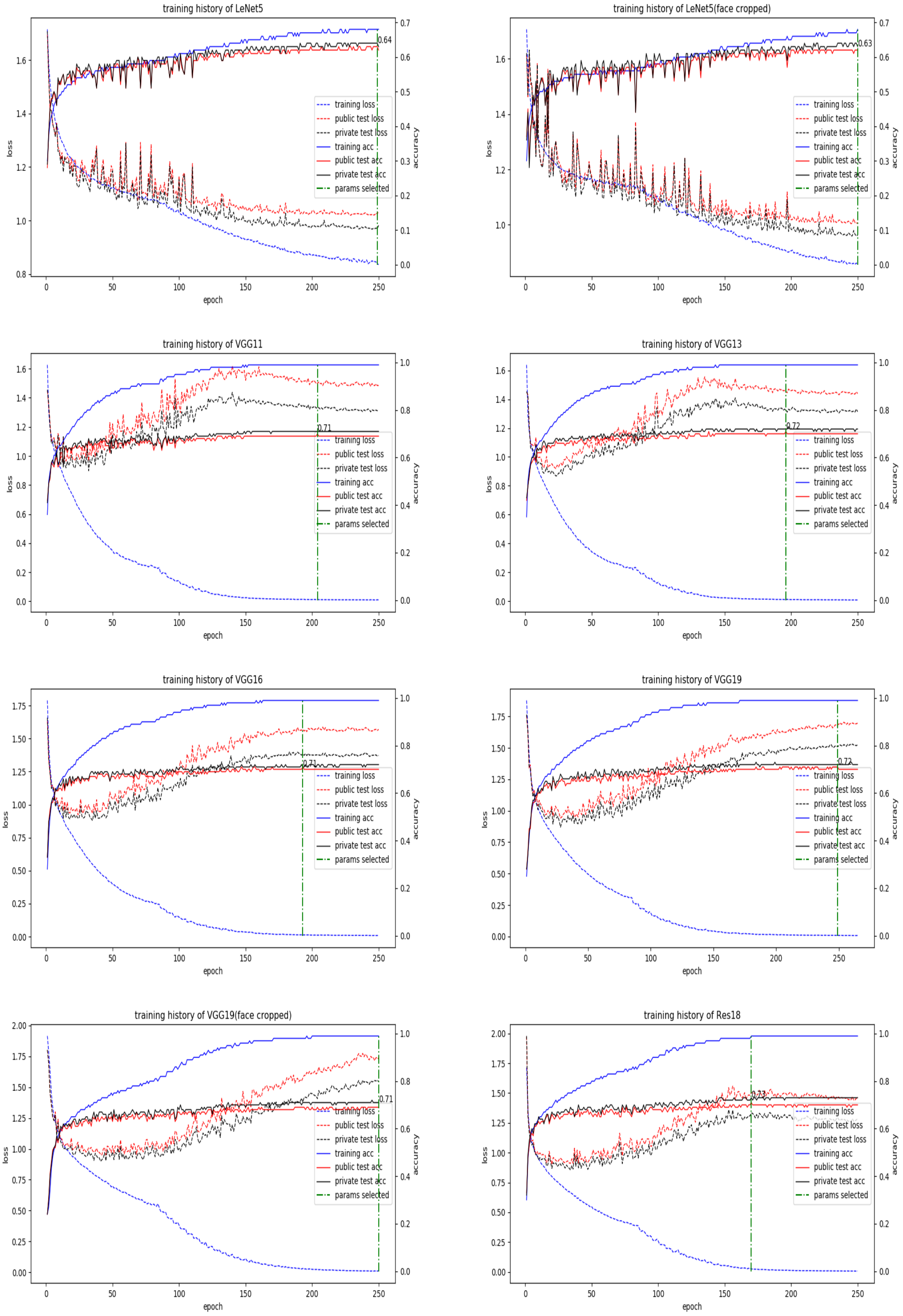

Fig. 5: Visualization of Training Histories