ESE 546, FALL 2023

HOMEWORK 2

KEQI WU [KEQIWU@SEAS.UPENN.EDU],
COLLABORATORS: NONE

Solution 1 (Time spent: 2 hours)**.** Your solution goes here.

(a)

1. Residual Network has residual blocks that have skip connections. The output of previous layer is added to the output of the layer after it. This process will change the dimension of input during convolution process. Skip connections can help to learn more complex data, provide a better generalization and reduce the number of parameters needed for training, making it easier and faster to train a deeper network.
2. Batch Normalization is applied before ReLU function. This helps to control the stability of the input. However, BN applied before ReLU should yield a better result as it has full control of the input. On the other hand, BN applied after reLU can not fully control the mean and variance of the input going into the activation function, which changes the statistics of the original input.
3. ResNet has bottleneck residual blocks. It applied a 1x1 convolution layer to reduces the number of channels in the input before major 3x3 convolution layer, then applied another 1x1 convlution layer to restore the number of channels to its orignal number. It helps to reduce computational complexity , preserve capacity of functions and improve efficiency.

(b)

model.train() switches the model into training mode. In the training mode, dropout layers are switched on. setting part of activations to zero. Batch normalization computes mean and variance over a mini-batch. On the other hand, model.eval() switches the model into evaluation mode. In the evaluation mode, there will be no dropout layers. Batch normalization takes mean and variance from the learned model. In the HW1, the neural network we implemented runs training and validation at the same time, so there won't be any difference between two modes. In our network, batch normalization and dropout layers are not implemented. We do not switch modes between training and testing. In order to add those features, we can set a probability of dropout rate and randomly sample input and hidden layers to set to 0. Batch normalization can be implemented by calculating mean and variance of mini batch in the training part, and validation part takes the same statistics for evaluation.

(c)

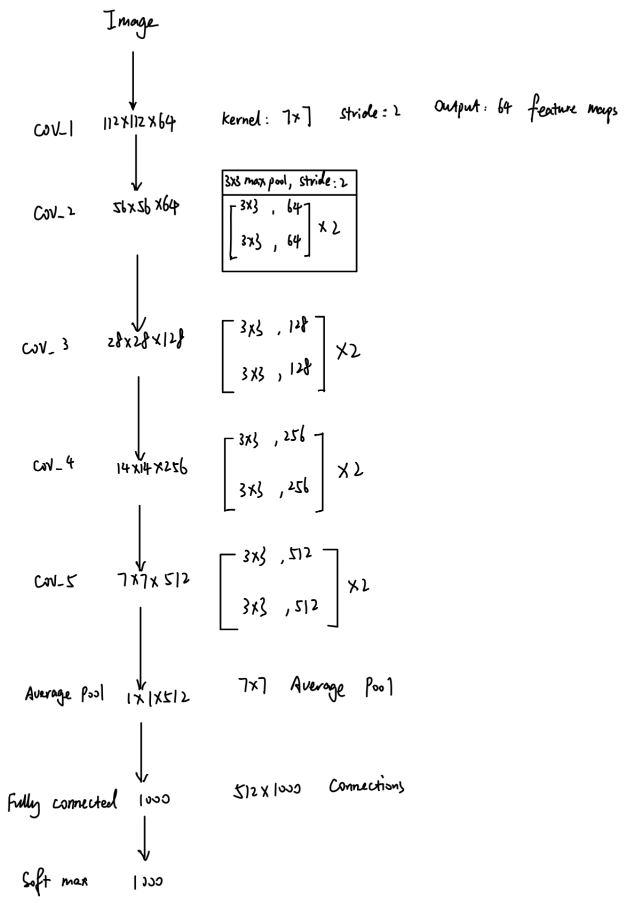
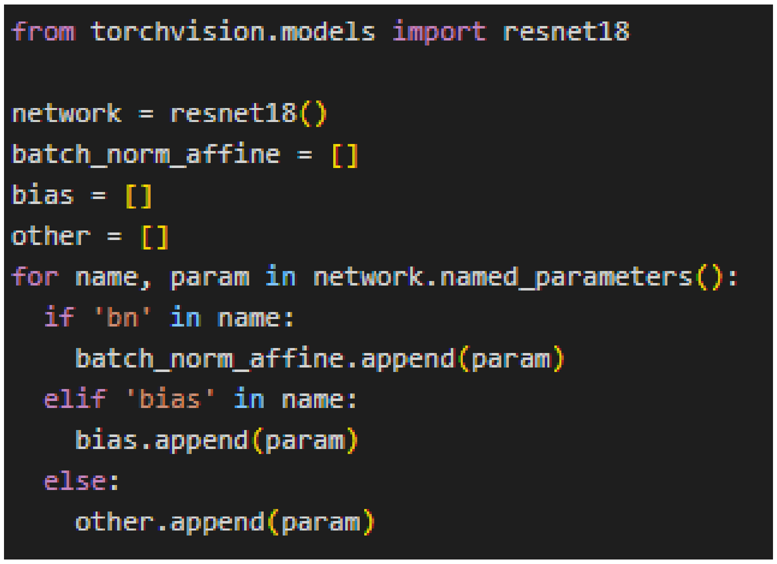


Image
COV_1 112x112x64
Kernel: 7x7 stride: 2 Output: 64 feature maps
COV_2 56x56x64
3x3 max pool, stride: 2
3x3 , 64
3x3 , 64
x2
COV_3 28x28x128
3x3 , 128
3x3 , 128
x2
COV_4 14x14x256
3x3 , 256
3x3 , 256
x2
COV_5 7x7x512
3x3 , 512
3x3 , 512
x2
Average Pool 1x1x512
7x7 Average Pool
Fully connected 1000
512x1000 Connections
Soft max 1000

(d)

Biases shift activation function to fit the data. It does not contribute too much to the flexibility of the model. Apply weight decay will potentially shift activation function inaccurately, making it less likely to perform well. The weight decay penalty prevents the bias parameter to adapt to the non-zero mean of the data. Applying weight decay will increase computational cost, which will decrease the training efficiency.

(e)



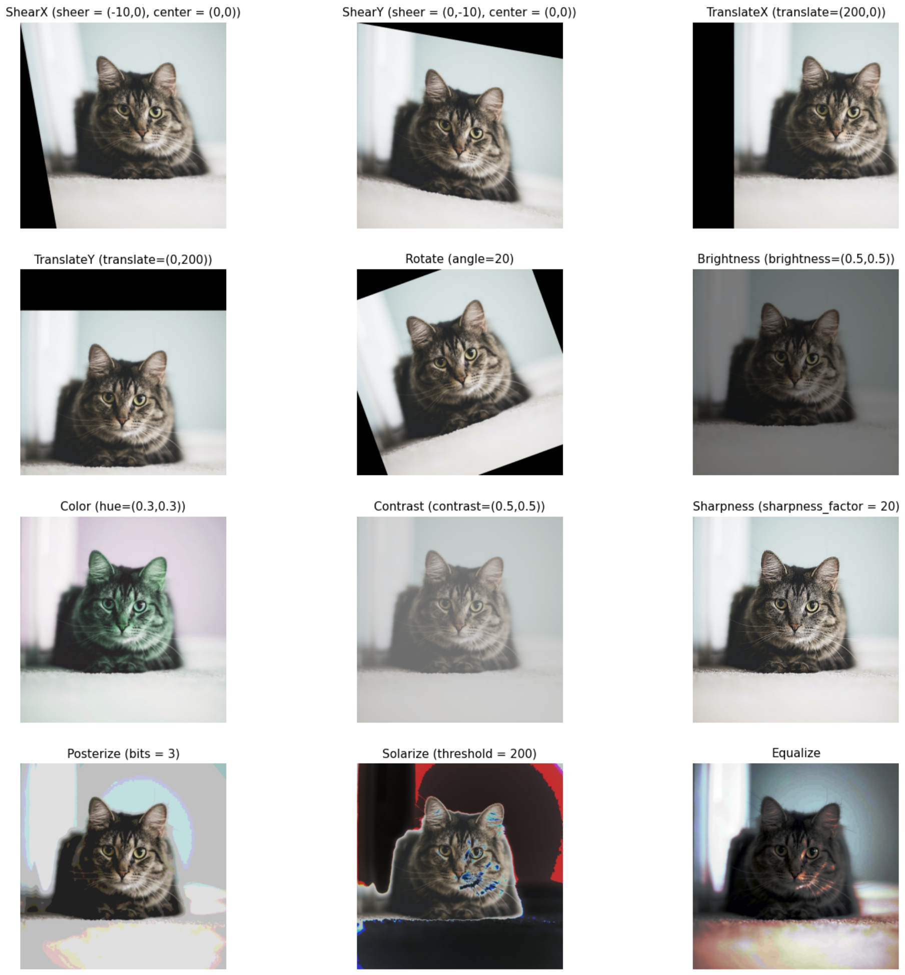
```python
from torchvision.models import resnet18

network = resnet18()
batch_norm_affine = []
bias = []
other = []
for name, param in network.named_parameters():
  if 'bn' in name:
    batch_norm_affine.append(param)
  elif 'bias' in name:
    bias.append(param)
  else:
    other.append(param)
```

(f)

Original Image:

Transformed:


ShearX (sheer = (-10,0), center = (0,0))
ShearY (sheer = (0,-10), center = (0,0))
TranslateX (translate=(200,0))
TranslateY (translate=(0,200))
Rotate (angle=20)
Brightness (brightness=(0.5,0.5))
Color (hue=(0.3,0.3))
Contrast (contrast=(0.5,0.5))
Sharpness (sharpness_factor = 20)
Posterize (bits = 3)
Solarize (threshold = 200)
Equalize

Solution 2 (Time spent: 1 hour)**.** Your solution goes here.

(a)

We can discuss the convexity using the second derivative:
let $f(A, B) = ||X - AB||_F^2$, then the first derivative is:

$$\begin{bmatrix} -2(X - AB)B^T \\ -2A^T(X - AB) \end{bmatrix}$$

The second derivative, or the Hessian Matrix is:

$$\begin{bmatrix} 2B^2 & -2X + 4AB \\ -2X + 4AB & 2A^2 \end{bmatrix}$$

We can guarantee that $2A^2$,$2B^2 \geq 0$, however, since there are no constraints enforced on three matrices, there is no evidence that $-2X + 4AB \geq 0$. We can not conclude that the Hessian Matrix is greater than or equal to 0. The second-order condition for convexity fails. Therefore, this objective function is non convex. This makes sense because the objective function is quadratic, which is not linear. There might exists multiple solutions that achieves the minimum.

(b)

Minimization Problem:

$$\min_{A\in\mathbb{R}^{m\times r},B\in\mathbb{R}^{r\times n}} f(A,B) = \min_{A\in\mathbb{R}^{m\times r},B\in\mathbb{R}^{r\times n}} ||X-AB||_F^2$$

Take SVD of X, $X = U\Sigma V^T$, where:

$$\begin{aligned}&\text{Orthogonal Matrix } U \in \mathbb{R}^{m\times n},\\&\text{Diagonal Matrix } \Sigma \in \mathbb{R}^{m\times n} \text{ with singular values,}\\&\text{Orthogonal Matrix } V^T \in \mathbb{R}^{n\times n}\end{aligned}$$

Then we rewrite minimization problem as:

$$\min_{A\in\mathbb{R}^{m\times r},B\in\mathbb{R}^{r\times n}} ||U\Sigma V^T - AB||_F^2$$

We want to capture different patterns with A and decide patterns to collect together with B, and we want them to be reduced to rank r.
Therefore, we choose A as the first r columns of U, or $A^* = U_{m\times r}$, and B^* as the first r rows of ΣV^T, or $B = (\Sigma V^T)_{r\times n}$

(c)

The solution is not unique as we said in part (a). First, we apply some linear transformation to matrix A^*, $C = A^*H$ with an invertible transformation matrix $H \in \mathbb{R}^{r\times r}$.

Accordingly, we apply linear transformation to $B*$ with inverse of H, $D = H^{-1}B^*$.

Then we have $CD = A^*HH^{-1}B^* = A^*B^*$, and we will achieve the same minimum value of the objective function.

In this case, we have:

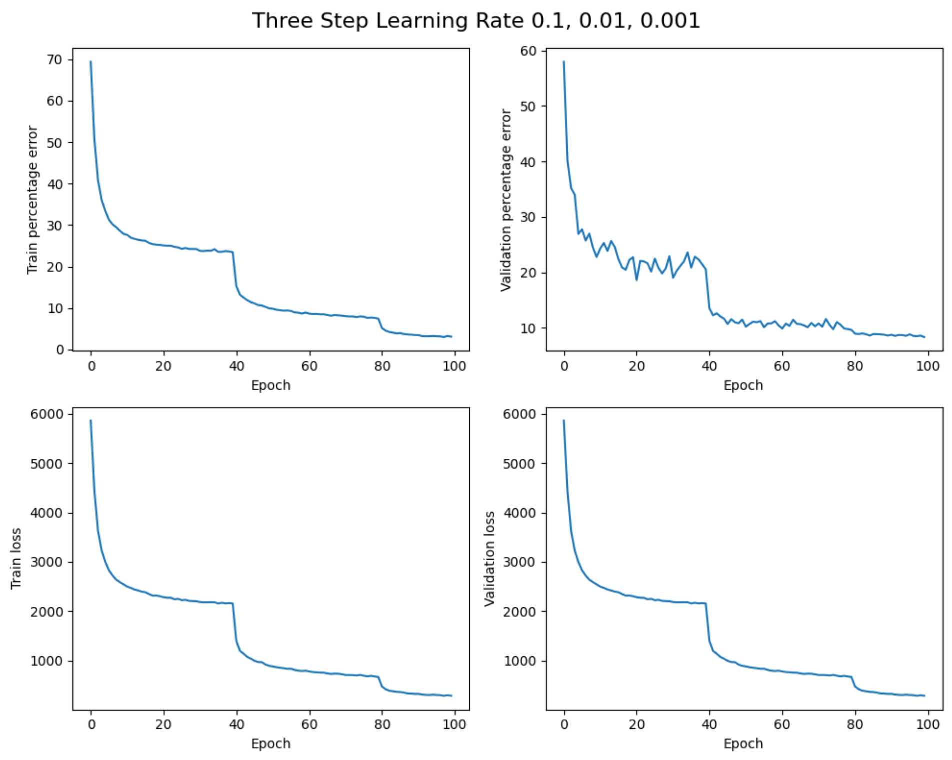
$$C = A^*H = U_{m\times r}H$$
$$D = H^{-1}B^* = H^{-1}(\Sigma V^T)_{r\times n}$$

We find another distinct solutions. Therefore, the solution to the above optimization problem is not unique

Solution 3 (Time spent: 5 hours)**.** Your solution goes here.

(a)

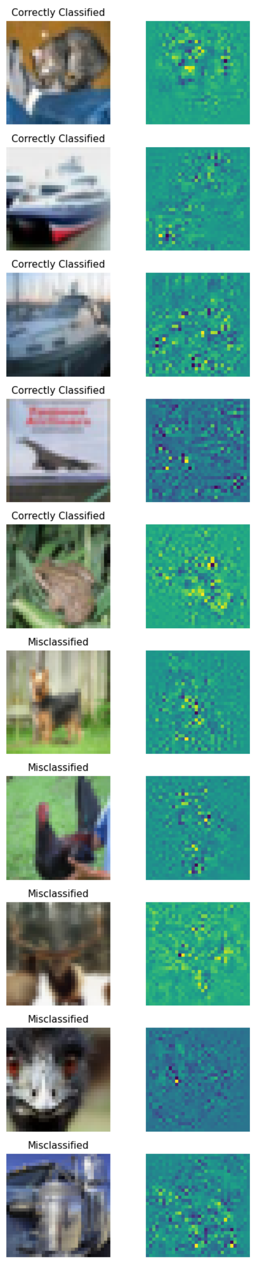
check code



(b)

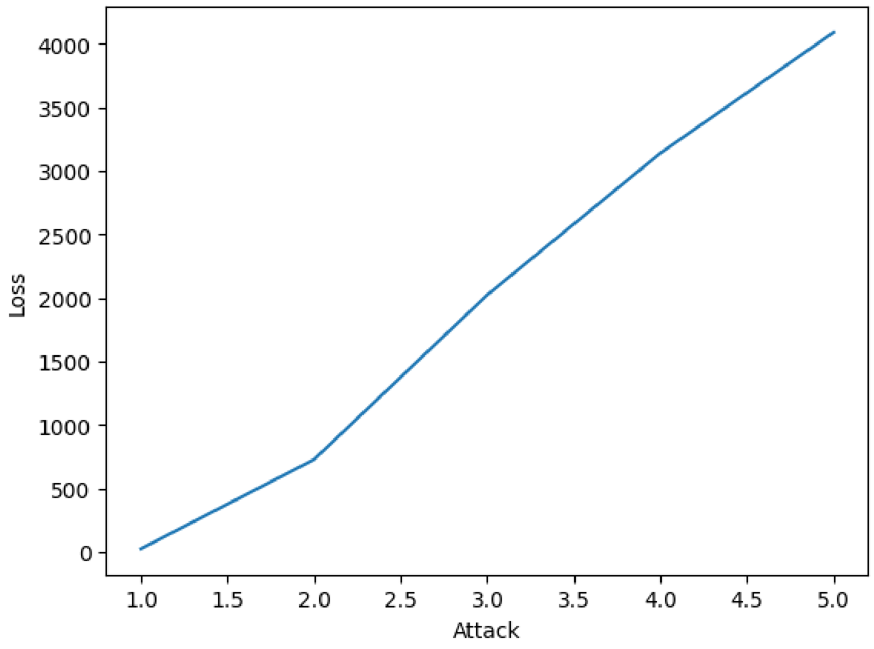
check code

Gradients of correctly classified images and misclassified images both captures the important features in images. Gradients detect edges and identify where intense changes of pixel values happening. However, for misclassified images, the models can not tell exactly which classes they belong to as the features captured by gradient is not good enough for classification



check code

Loss for 5 way attack:



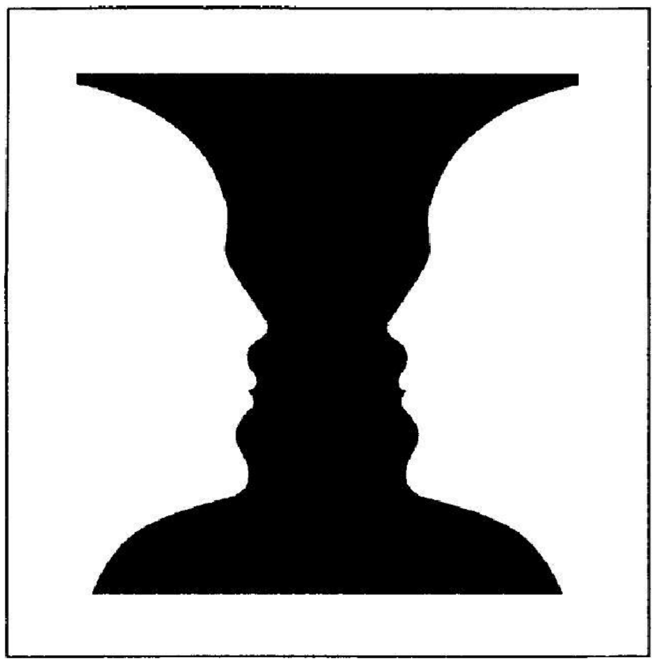
(c)

check code

Accuracy of the network before attack: 91.7 %
Accuracy of the network after attack: 25.64%

(d)

Yes, humans are susceptible to adversarial perturbations. When looking at the images, every representation of pixel is influenced by pixels around it. Our brain system makes adjustment automatically and controls what we see. For example, rubin's vase:



We can classify it as ”a vase” or ”two faces”. With a small perturbations (add some noise), human can not distinguish them clearly, and they are likely to misclassify it. Therefore, humans are susceptible to even small perturbations

Solution 4 (Time spent: 1 hours)**.** Your solution goes here.

(a)

$$\begin{aligned}\frac{\partial l(z^T)}{\partial w_z} &= \frac{\partial l(z^T)}{\partial z^T}(\prod_{i=1}^{T-1}\frac{\partial z^{j+1}}{\partial z^j})\frac{\partial z^1}{\partial w_z}\\ &= \frac{\partial l(z^T)}{\partial z^T}(\prod_{i=1}^{T-1} w_z)\frac{\partial z^1}{\partial w_z}\end{aligned}$$

Then,

$$\begin{aligned}||\frac{\partial l(z^T)}{\partial w_z}||_2 &= ||\frac{\partial l(z^T)}{\partial z^T}(\prod_{i=1}^{T-1} w_z)\frac{\partial z^1}{\partial w_z}||_2\\ &\leq ||\frac{\partial l(z^T)}{\partial z^T}||_2 \cdot ||(\prod_{i=1}^{T-1} w_z)||_2 \cdot ||\frac{\partial z^1}{\partial w_z}||_2\end{aligned}$$

In order to let this inequality happen, we need to have $||(\prod_{j=i}^{T-1} w_z)||_2 < 1$. Therefore, when $||w_z||_2 < 1$, the gradient is vanishing. when $||w_z||_2 > 1$, the gradient increases and eventually explodes.

(b)

We have a linear activation function, so it can easily explode. If we want to use nonlinear activation function, we need to make sure that it is bounded. An unbounded activation function have no restriction on the output. With steep curve, it can yield extremely large value. Therefore, we should choose bounded nonlinear activation function, for example, sigmoid and tanh, so we can bound output in [-1,1]

(c)

To prevent exploding gradient, we can apply gradient clipping by scaling gradient down if the norm of gradient is above some threshold:

$$w_z^{new} = w_z^{old} - \eta * clipping(\frac{dl}{dw_z}, threshold)$$

To prevent vanishing gradient, we can apply regularization or weight decay to add a small bias to the update, let λ be the regularization parameter:

$$w_z^{new} = w_z^{old} - \eta * \frac{dl}{dw_z} - \eta * \lambda * w_z^{old}$$

(d)

LSTM applies gating mechanisms. By adjusting the forget gate's parameters at each time step, the network is able to more effectively manage the gradient values since the gradient stores the activation vector of the forget gate. The LSTM can decide whether or not to recall specific information at each time step by activating the forget gate, and then adjusting the model's parameters.

(e)

Self-attention can adds context-related information to an input, thus improving its information richness. Also, the self-attention method can capture dependencies between any two data points in the sequence, which allows the model to dynamically modify the impact of each element on the output based on how important it is to each component in an input sequence.