CS 446 MJT — Homework 2

your NetID here

Version 1

Instructions.

- Homework is due **Tuesday, February 26, at 11:59pm**; no late homework accepted. You will have everything you need to solve these problems after both deep network lectures.
- Everyone must submit individually at gradescope under `hw2` and `hw2code`.
- The "written" submission at `hw2` **must be typed**, and submitted in any format gradescope accepts (to be safe, submit a PDF). You may use LaTeX, markdown, google docs, MS word, whatever you like; but it must be typed!
- When submitting at `hw2`, gradescope will ask you to mark out boxes around each of your answers; please do this precisely!
- Please make sure your NetID is clear and large on the first page of the homework.
- Your solution **must** be written in your own words. Please see the course webpage for full academic integrity information. Briefly, you may have high-level discussions with at most 3 classmates, whose NetIDs you should place on the first page of your solutions, and you should cite any external reference you use; despite all this, your solution must be written in your own words.
- We reserve the right to reduce the auto-graded score for `hw2code` if we detect funny business (e.g., rather than implementing an algorithm, you keep re-submitting the assignment to the auto-grader, eventually completing a binary search for the answers).
- There are **no regrade requests** on `hw2code`, which is the code auto-grader; however, you can re-submit and re-grade as many times as you like before the deadline! Start early and report any issues on piazza!
- Methods and functions in the template and utility code include docstrings to describe the inputs and outputs. The autograder relies on correct implementations of these methods. Follow the docstrings to avoid failing tests.

1. **Singular Value Decomposition and norms.**

   (a) Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, define its *spectral norm* $\|\boldsymbol{A}\|_2$ as

   $$\|\boldsymbol{A}\|_2 := \max_{x \in \mathbb{R}^d, \|x\|_2 \leq 1} \|\boldsymbol{A}\boldsymbol{x}\|_2.$$

   Prove that $\|\boldsymbol{A}\|_2$ is equal to the largest singular value of $\boldsymbol{A}$. You may assume $\boldsymbol{A}$ is not the 0 matrix.

   Hint: write $\boldsymbol{A}$ in terms of its SVD, and rewrite $\boldsymbol{x}$ in one of the bases provided by the SVD.

   (b) Prove that for any matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ and vector $\boldsymbol{x} \in \mathbb{R}^d$, then $\|\boldsymbol{A}\boldsymbol{x}\|_2 \leq \|\boldsymbol{A}\|_2 \cdot \|\boldsymbol{x}\|_2$.

   (c) Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, define its *Frobenius norm* $\|\boldsymbol{A}\|_2$ as

   $$\|\boldsymbol{A}\|_{\text{F}} = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{d} A_{ij}^2}.$$

   (This is the same as the vector $\|\cdot\|_2$ norm of the unrolled/flattened/vectorized matrix.)
   Prove that $\|\boldsymbol{A}\|_{\text{F}}^2 = \text{tr}(\boldsymbol{A}^\intercal \boldsymbol{A})$, where tr denotes the trace.

   (d) Continuing with the previous part, now show

   $$\|\boldsymbol{A}\|_{\text{F}} = \sqrt{\sum_{i=1}^{r} s_i^2},$$

   the (vector) ℓ_2 norm of the singular values of $\boldsymbol{A}$ (or 0 if there are none).
   Hint: the previous part gives a few convenient ways to do this problem after you replace $\boldsymbol{A}$ with its SVD.

   (e) Given matrices $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{B} \in \mathbb{R}^{d \times m}$, prove

   $$\|\boldsymbol{A}\boldsymbol{B}\|_{\text{F}} \leq \|\boldsymbol{A}\|_2 \cdot \|\boldsymbol{B}\|_{\text{F}}.$$

   Hint: note that $\|\boldsymbol{B}\|_{\text{F}}^2 = \sum_{i=1}^{m} \|\boldsymbol{B}_{:,i}\|_2^2$, where $\boldsymbol{B}_{:,i}$ denotes the i^{th} column of $\boldsymbol{B}$.

   (f) Suppose $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ has rank $r \geq 1$. Prove

   $$\|\boldsymbol{A}^+\boldsymbol{A}\|_2 = \|\boldsymbol{A}\boldsymbol{A}^+\|_2 = 1$$

   and

   $$\|\boldsymbol{A}^+\boldsymbol{A}\|_{\text{F}} = \|\boldsymbol{A}\boldsymbol{A}^+\|_{\text{F}} = \sqrt{r}.$$

   (g) Choose a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ with rank $1 \leq r < n$, where $n \geq 2$, and a vector $\boldsymbol{v} \in \mathbb{R}^n$ so that $\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{v} = 0$ but $\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{v} \neq 0$.
   Please use only 1-2 sentences for your answer.
   Hint: one convenient way to solve and then state your answer is to define/construct $\boldsymbol{A}$ via its SVD.
   Remark: the point of this part is that inverses and pseudoinverses really can behave differently!

Solution. *(Your solution here.)*

(a) **Proof:** Thin SVD of A, $A = USV^T$. Then rewrite x in terms of V, $x = Vy$.

Note that $||x||_2 = x^T x = y^T V^T V y = ||y||_2 \leq 1$. So we have

$$||Ax||_2 = ||USV^T Vy||_2 = ||USy||_2 = \sqrt{y^T S^T S y}$$

Assume the singular values of A are $s_i, 0 < i \leq r$, then we have

$$\sqrt{y^T S^T S y} = \sqrt{\sum_{i=1}^{r} s_i^2 y_i^2}$$

$$s.t. \sum_{i=1}^{r} y_i^2 \leq 1$$

So $\max_{||x||_2 \leq 1} ||Ax||_2 = \max s_i$.

(b) **Proof:**

$$||A||_2 = \max_{||x||_2 \leq 1} ||Ax||_2 \geq ||A\frac{x}{||x||_2}||_2 = \frac{||Ax||_2}{||x||_2}$$

Hence, we have $||A||_2 ||x||_2 \geq ||Ax||_2$

(c) **Proof:** Assume A = $\begin{bmatrix} a_1 & a_2 & \cdots & a_d \end{bmatrix}$ Then

$$A^T A = \begin{bmatrix} a_1^T a_1 & a_1^T a_2 & \cdots & a_1^T a_d \\ a_2^T a_1 & a_2^T a_2 & \cdots & a_2^T a_d \\ \vdots & \vdots & \ddots & \vdots \\ a_d^T a_1 & a_d^T a_2 & \cdots & a_d^T a_d \end{bmatrix}$$

$$tr(A^T A) = \sum_{i=0}^{d} a_i^T a_i = \sum_{i=1}^{d} \sum_{j=1}^{d} A_{ij}^2$$

Q.E.D

(d) **Proof:** Write A as $A = USV^T$, then we have

$$tr(A^T A) = tr(VS^T U^T USV^T) = tr(VS^T SV^T) = \sum_{i=1}^{n} \sum_{j=1}^{r} s_j^2 v_{ij}^2 = \sum_{j=1}^{r} s_j^2 \sum_{i=1}^{n} v_{ij}^2 = \sum_{j=1}^{r} s_j^2$$

Hence,

$$||A||_F^2 = \sum_{j=1}^{r} s_j^2$$

(e) **Proof:** First, we proof that

$$\max_{||x||_2 \leq 1} ||ABx||_2 \leq ||A||_2||Bx||_2 \leq ||A||_2||B||_2||x||_2 = ||A||_2||B||_2$$

Then we have

$$||A||_2^2||B||_F^2 = \sum_{i=1}^{m} ||A||_2^2||B:,i||_2^2 \geq \sum_{i=1}^{m} ||AB;,i||_2^2 = ||AB||_F^2$$

(f) **Proof:** Thin SVD:

$$A = USV^T$$

Pseudo-inverse:

$$A^+ = VS^+U^T$$

Hence, we have

$$A^+A = VS^+U^TUSV^T = VV^T = VI_rV^T$$
$$AA^+ = USV^TVS^+U^T = UU^T = UI_rU^T$$

Hence,

$$||AA^+||_2 = ||UU^T||_2 = \max s_i = 1$$
$$A^+A||_2 = ||VV^T||_2 = \max s_i = 1 = ||AA^+||$$

We also have

$$||A^+A||_F = ||VV^T||_F = \sqrt{\sum_{i=1}^{r} s_i} = \sqrt{r} = ||AA^+||_F$$

(g) **Proof:** Assume $n \times r$ orthogonal matrix V and U, and the vector v satisfies $V^TX = 0$ and $U^Tx \neq 0$, then construct $A = UI_rV^T$.

2. **Singular Value Decomposition and image reconstruction.**

   (a) Say that if $\boldsymbol{A} = \sum_{i=1}^{r} s_i \boldsymbol{u}_i \boldsymbol{v}_i^\intercal$, then $\sum_{i=1}^{\min\{k,r\}} s_i \boldsymbol{u}_i \boldsymbol{v}_i^\intercal$ is the *k-reconstruction* of $\boldsymbol{A}$. Similarly, we define the *min-k-reconstruction* to be $\sum_{i=\max\{1,r-k+1\}}^{r} s_i \boldsymbol{u}_i \boldsymbol{v}_i^\intercal$. Implement `reconstruct_SVD`, as per the docstrings.

   (b) Add in your writeup a log-scale plot of the singular values of the red color channel: Let $\boldsymbol{A}$ denote the red color channel and $\boldsymbol{A} = \sum_{i=1}^{r} s_i \boldsymbol{u}_i \boldsymbol{v}_i^\intercal$ denote its singular value decomposition, for each $1 \leq i \leq r$, plot $\ln(1 + s_i)$. Comment on the distribution of the singular values. Is the log-singular-value plot linear?

   (c) Use the function `get_img` to get an old NASA Astronomy Picture of the Day. Include in your report plots of

   i. The original image
   ii. The 100-reconstruction of the image
   iii. The 50-reconstruction of the image
   iv. The 20-reconstruction of the image
   v. The 10-reconstruction of the image
   vi. The min-600-reconstruction of the image

Solution. *(Your solution here.)*

(a)

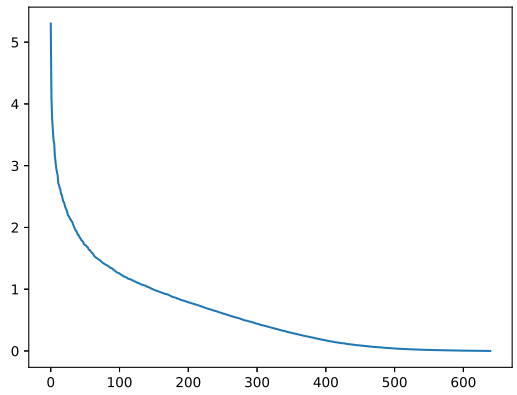

Figure 1: Log-singular-value plot

(b) The log-singular-value plot is not l
near. Only a little portion of the singular values take up most of the weight.

(c)

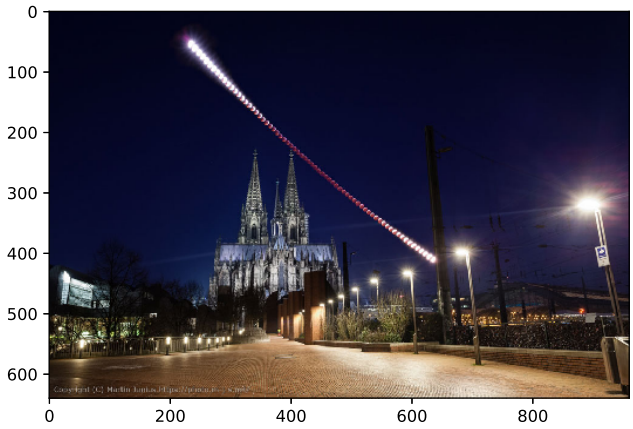

Figure 2: Original image

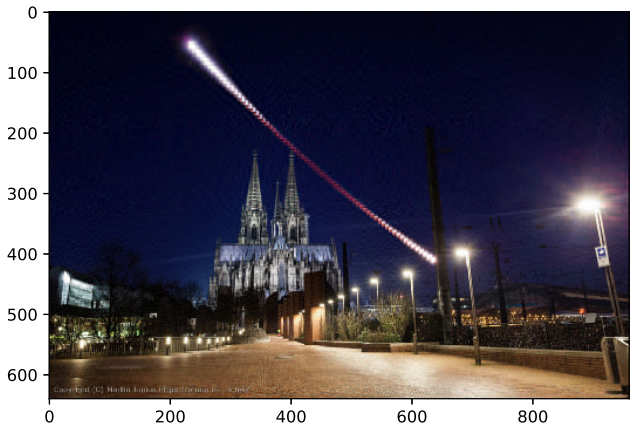

Figure 3: The 100-reconstruction of the image

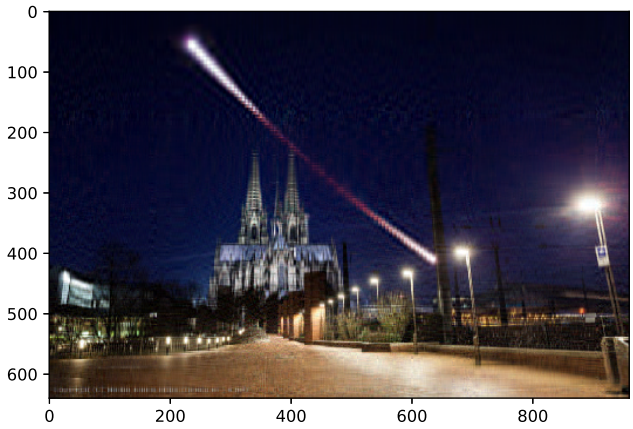

Figure 4: The 50-reconstruction of the image

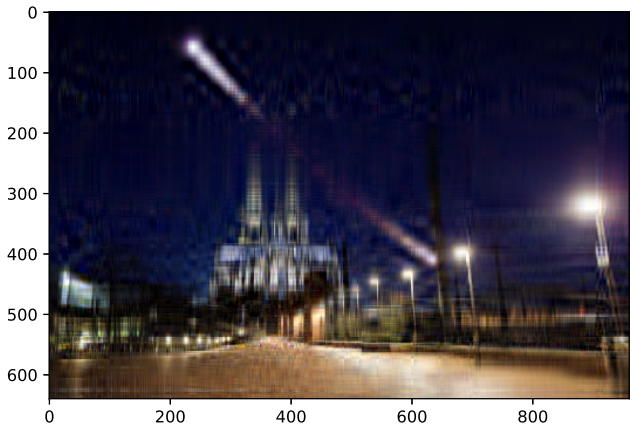

Figure 5: The 20-reconstruction of the image

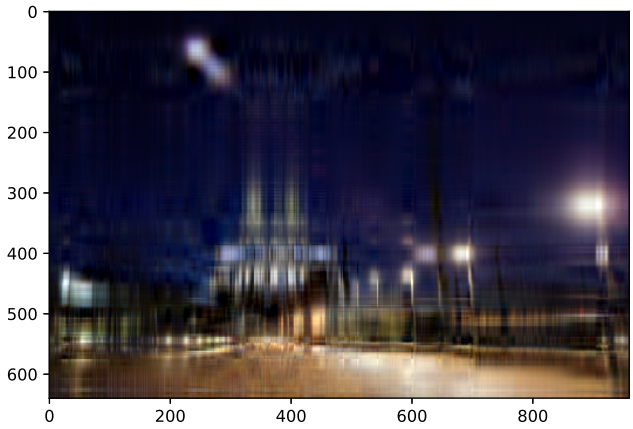

Figure 6: The 10-reconstruction of the image

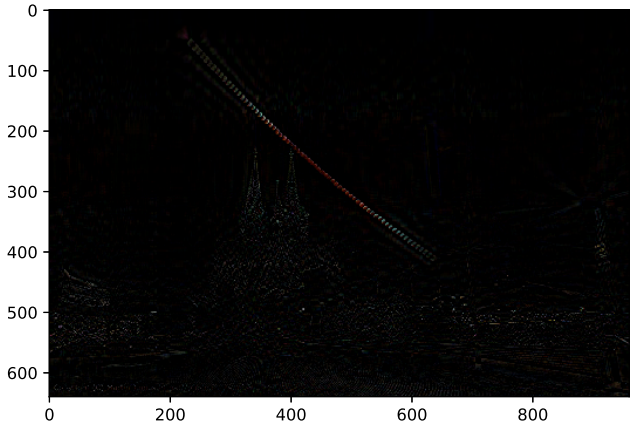

Figure 7: The min-600-reconstruction of the image

3. **Neural Networks on XOR.**

In this problem you will demonstrate that a two-layer neural network with the ReLU activation function can classify the XOR dataset correctly. This will also serve as an introduction to writing your own neural networks in PyTorch! Consider the two layer neural network below

$$\boldsymbol{x} \mapsto \boldsymbol{W}_2\sigma_1(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2.$$

(a) Implement your network in the class XORNet. You will need to modify `__init__`, `set_l1`, `set_l2`, and `forward` methods. The setter methods are used by the autograder to test your network implementation. *Note*: to maintain consistency with PyTorch's `torch.nn.Linear`, the arguments to `set_l1`, `set_l2` will have shapes consistent with the following network formulation: $f(\boldsymbol{X}) = \sigma_1(\boldsymbol{X}\boldsymbol{W}_1^\intercal + \boldsymbol{b}_1)\boldsymbol{W}_2^\intercal + \boldsymbol{b}_2$, where $\boldsymbol{X} \in \mathbb{R}^{n\times d}$.

(b) Define the `fit` function. Please refer to the docstring and template code for details.

(c) Using your `fit` function, train an XORNet on the XOR dataset for 5000 epochs, and then use `contour_torch` to plot your resulting network. Include the plot in your writeup. Did you successfully classify the XOR points, or did your gradient descent get stuck in a local minima of the loss function?

Solution. *(Your solution here.)*

(a)

(b)

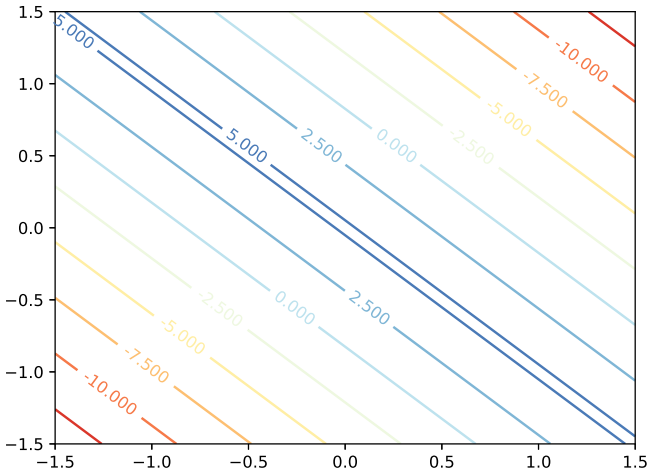

Figure 8: Fit function

(c) From the contour plot, we can see the XOR points are successfully classified.
I use SGD and it didn't stuck into a local minima.
But I did notice that if the step size is too small (like 0.01), 5000 steps are not enough for it to reach its global minima.

4. **Convolutional Neural Networks.**

In this problem, you will use convolutional neural networks to learn to classify handwritten digits. The digits will be encoded as 8x8 matrices. The layers of your neural network should be:

- A 2D convolutional layer with 1 input channel and 8 output channels, with a kernel size of 3
- A 2D maximimum pooling layer, with kernel size 2
- A 2D convolutional layer with 8 input channels and 4 output channels, with a kernel size of 3
- A fully connected (torch.nn.Linear) layer with 4 inputs and 10 outputs

Apply the ReLU activation function to the output of each of your convolutional layers before inputting them to your next layer. For both of the convolutional layers of the network, use the default settings parameters (stride=1, padding=0, dilation=1, groups=1, bias=True).

(a) Implement the class `DigitsConvNet`. Please refer to the dosctrings in hw2.py for details.

(b) Implement `fit_and_validate` for use in the next several parts. Please do not shuffle the inputs when batching in this part! The utility function `loss_batch` will be useful. See the docstrings in hw2.py and hw2_util.py or details.

(c) Fit a `DigitsConvNet` on the train dataset from `torch_digits`. Use CrossEntropyLoss, an SGD optimizer with learning rate 0.005 and no momentum, and train your model for 30 epochs with batch size of 1. Keep track of your training and validation loss for the next part.

(d) Fit another `DigitsConvNet`. This time we will adjust the learning rate so that it decreases at each epoch. Recall the gradient descent update step

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \eta_t \nabla_{\boldsymbol{w}} F(\boldsymbol{w}_i).$$

Here, i is the step, and t is the epoch. We will update the learning rate at each epoch so $\eta_{t+1} = \gamma \eta_t$. You should use `torch.optim.lr_scheduler.ExponentialLR`. We will use the a decay rate of $\gamma = 0.95$, and start the learning rate at 0.005. Save your neural network to a file using `torch.save(model.cpu().state_dict(), "conv.pb")`. You will submit this file to the autograder.

(e) Fit a third `DigitsConvNet`, again with an SGD optimizer with learning rate 0.005 and no momentum. However, this time, use a batch size of 16. Plot the epochs vs loss for parts (b), (c), and (d). Include the plot and your assessment of the impact of the exponentially decayed learning rate on training speed. Additionally, comment on the impact of increasing batch size. (In your report, may simply leave parts (c) and (d) blank, and include all comments in part (e).)

(f) The last layer of the network can be interpreted as a linear classifier on top of a *feature representation* constructed by the earlier layers. The purpose of this sub-question is to assess the quality of these features. Implement the method `intermediate`, as specified in the docstring. Then, use the function `plot_PCA` (included in the hw2_utils.py) to make a scatterplot of your the intermediate representation of the training data. (You will learn about PCA later in the course, but for now, it is sufficient to know it can be used for dimensionality reduction.) Include your plot in the writeup. Use your best neural net (according to validation accuracy) for this plot.

(g) Use the feature representation of your training dataset and labels to create a `scipy.spatial.KDTree`. Then, do 5-nearest-neighbors classification of the the feature representation of your validation dataset with the KDTree (this uses the KDTree's `query` method). You should use the same neural net as in part (f) for this problem. Report the accuracy of your KDTree on the validation set.

Solution. *(Your solution here.)*

(a)

(b)

(c)

(d)

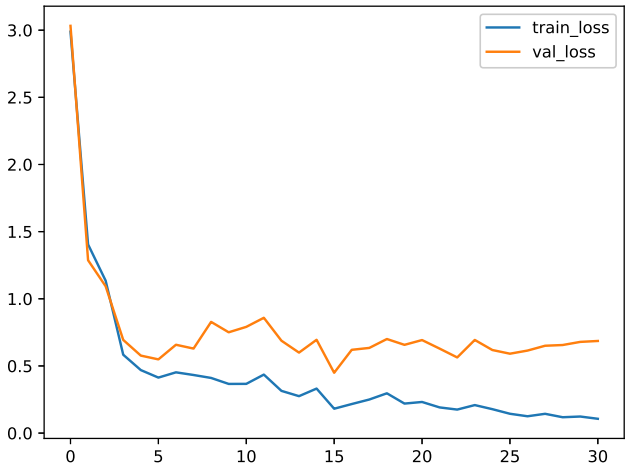

Figure 9: Loss for (c)

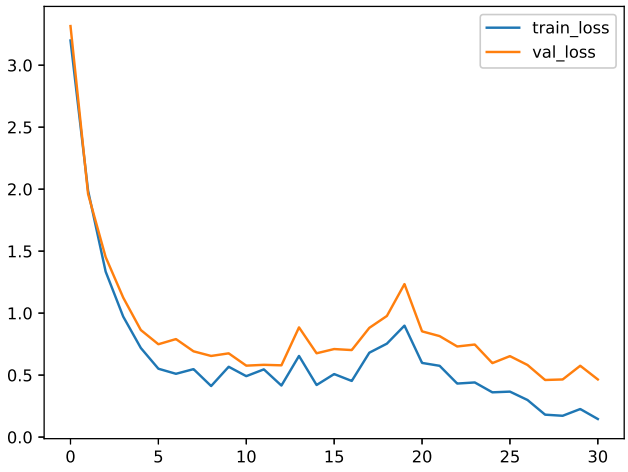

Figure 10: Loss for (d)

(e) By comparing (c) and (d), I find that the training loss decreases slower after applying exponentially decayed learning rate.
But the validation loss decreases faster. The overfitting seems disappeared.
When the batch size is set to 16, the loss curve is smoother. However, the training loss and validation loss decrease slower.

(f) See Figure 12.

(g) The accuracy is 0.88333.

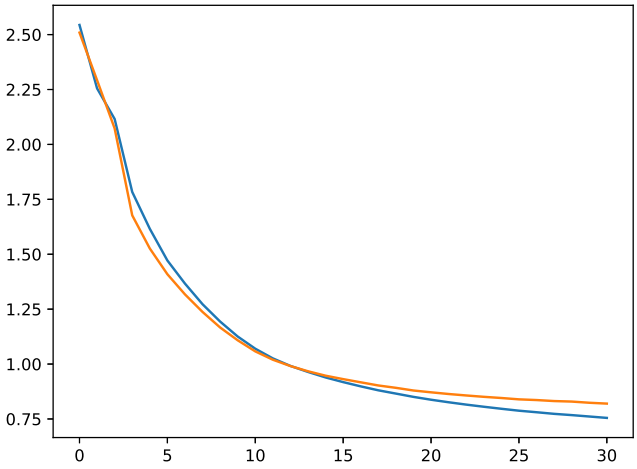

Figure 11: Loss for (e)

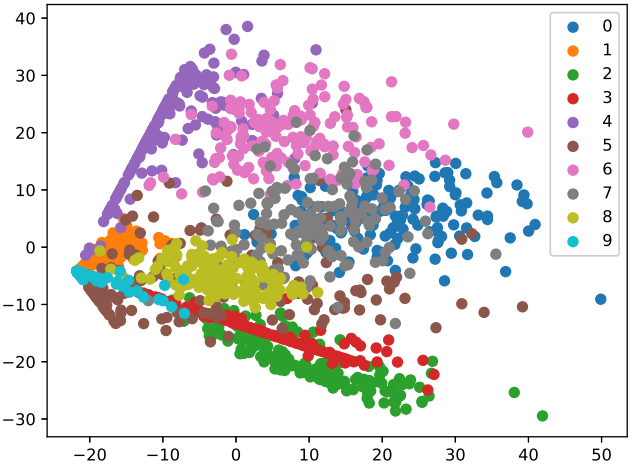

Figure 12: PCA plot