IDL EX2 Nadav Eisen and Yonatan Microshnik

June 30, 2024

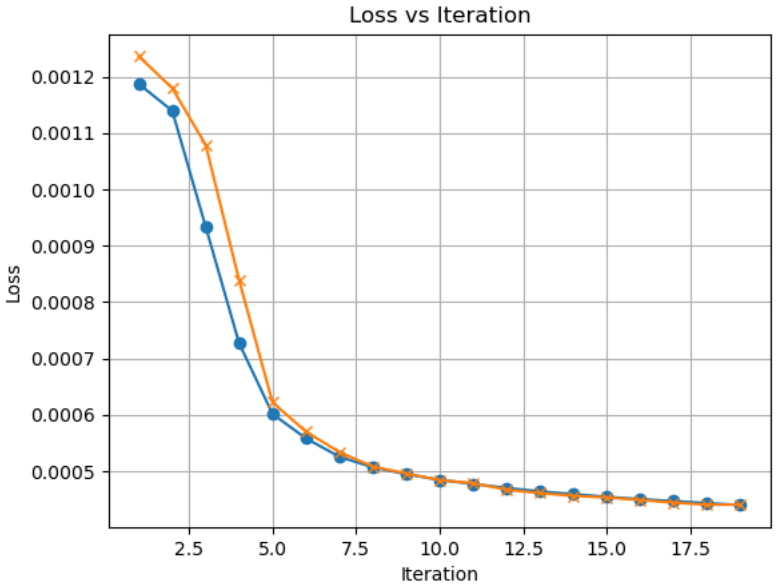
Practical Part:

Question 1:

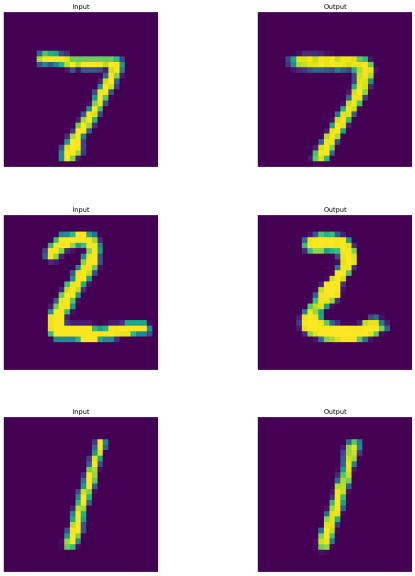
We construct the Encoder from 3 convolutional layers, each one having a 3x3 kernel, with stride of 2 and padding of 1, afterwhich we also use two Linear layers. We use non-linearity in the forward function by using ReLU on the output of each convolutional layer. The Decoder is the exact mirror image of the Encoder, as in, it begins with two Linear layers and ends with 3 convolutional layers.

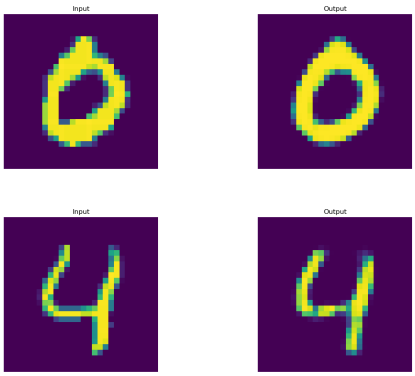
We chose this architecture because we knew from the lecture that images with clear patterns can be learned well by convolutional layers, and that adding the Linear layers can add another sort of logic to fully encode the convolutional representations.

We've run the training, and got a graph to show the loss over the training iteration:



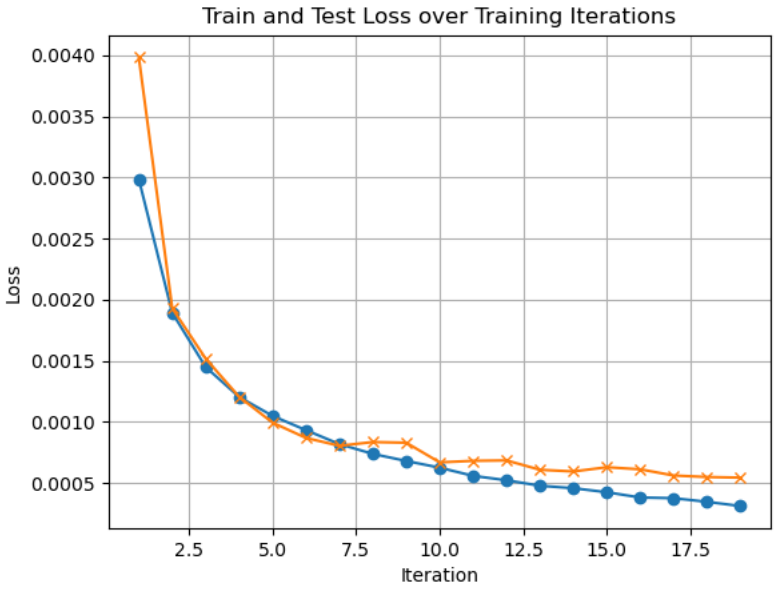
And we've also made several example outputs with the final trained model:

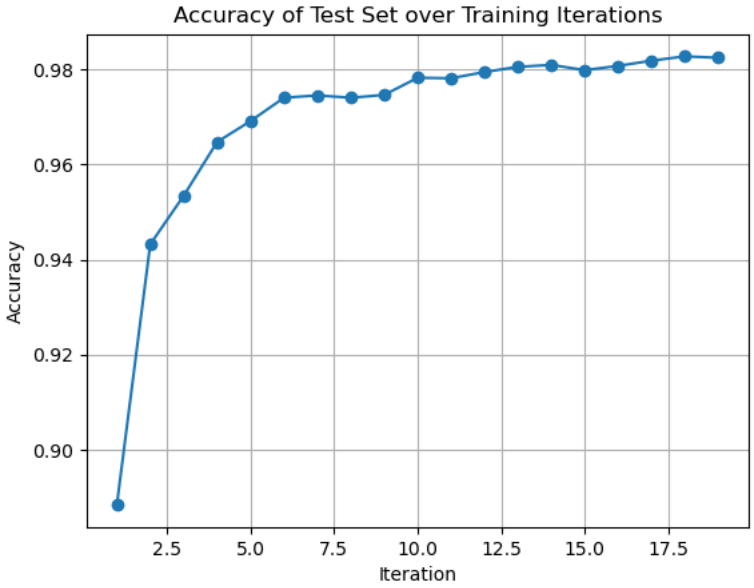




Question 2:

After training the Classifier, we got the following graphs:

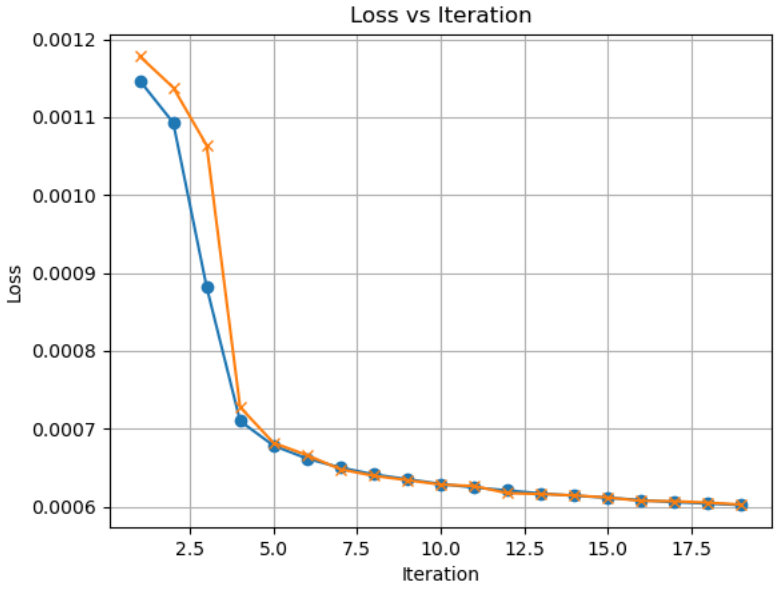


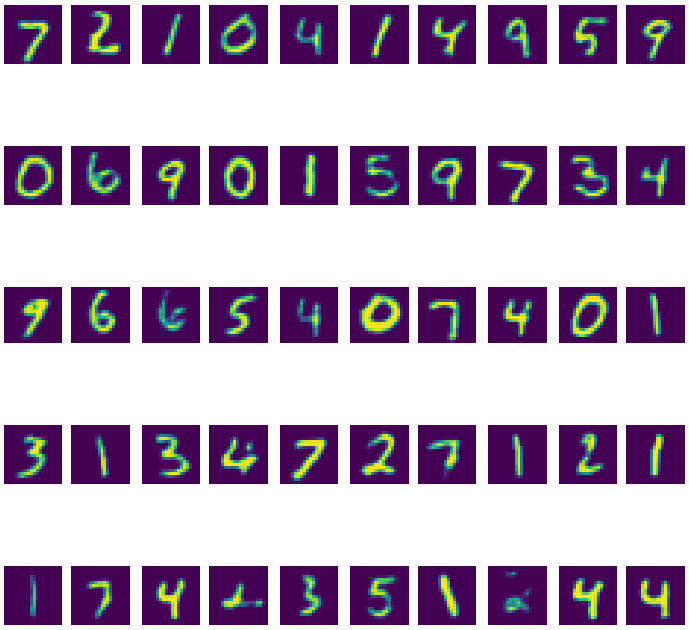


Question 3:

*)

The loss graph and the example outputs:


Loss vs Iteration
0.0012
0.0011
0.0010
0.0009
0.0008
0.0007
0.0006
Loss
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Iteration



a)

The question 1 Autoencoder's Encoder would encode a representation that would be useful for the later decoder to easily get the needed information for the particaular features of a given digit image. On the other hand, the Encoder of question 2 would represent a representation that would allow the creation of a 10-length vector whos values represent the probabilities(given by a softmax activation function at the end) that the given image is a certain class of digit.

b)

Per digit variability is higher in the question 1 autoencoder since it tries to create a feature space that represents imagery of a digit instead of the class of the image.

In question 3, because we use the classifier encoder from question 2, the representations that it creates only give the Decoder a vector of probabilities of

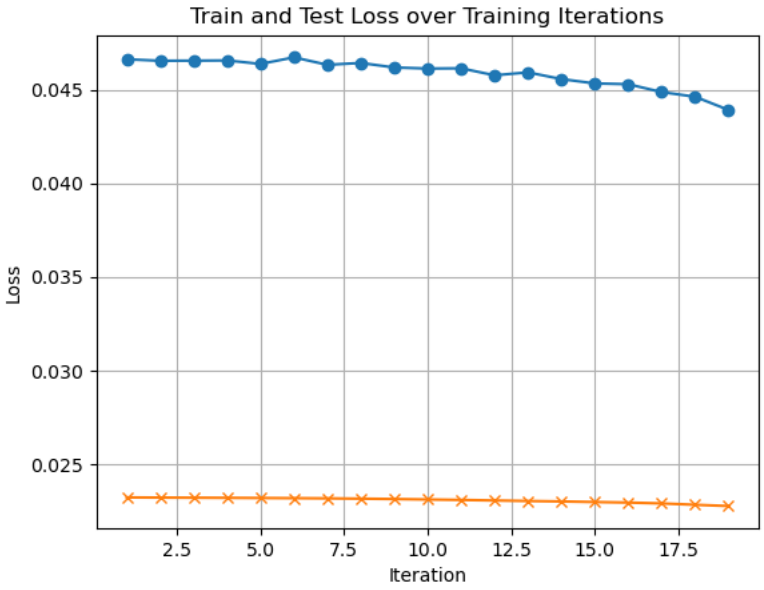
which class of digit it is, meaning that each decoder output is more representative of a general example of a digit than a specific digit(IE the input at this situation).

c)

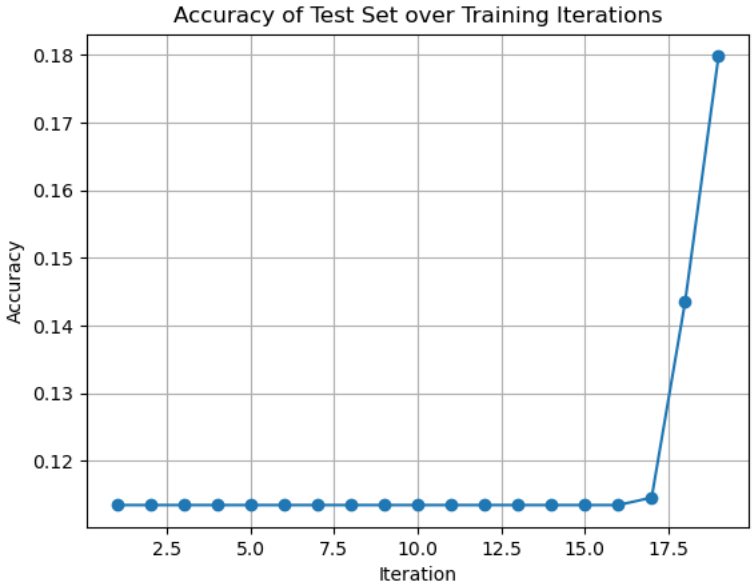
They are either as similar, or there is a slightly higher separation between digits in the question 3 autoencoder because the question 1 autoencoder also better represents the exact value input(the 0 to 1 intensity) and so the question 3 autoencoder creates images that represent digits in a worse and less consistent, and thus more heterogenous way than what we find in question 1.

Question 4:

For the few-examples models, after training the Classifier, we got the following graphs:



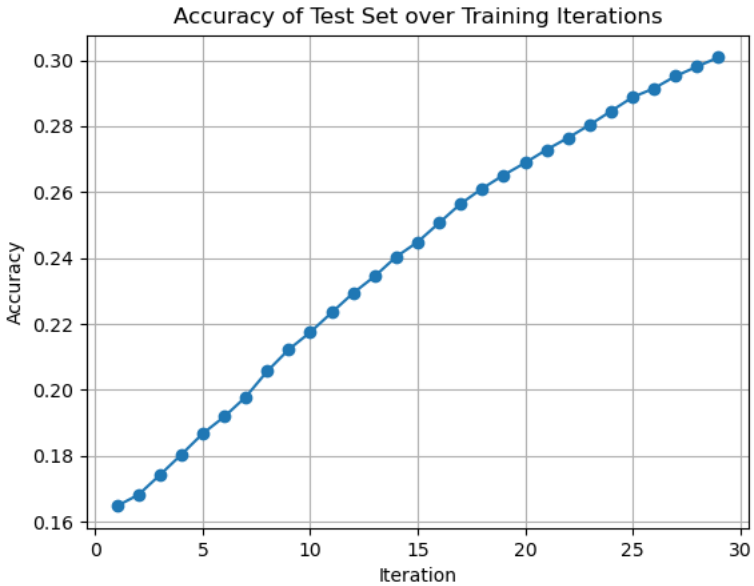
For the few-examples models, after training the Autoencoder, we got the following graphs:



Over-fitting is somewhat scene at the very end wherein we see that the model is able to get a small accuracy boost where we assume it learned the specific represantations of a few specific images.

Question 5:

For the transfer learning, after training the Classifier, we got the following graphs:



We can see that the trasfer learning allowed our few-examples dataset to still learn quite a bit more than the situation in the previous section, though a large amount of overfitting was seen because the weights already held quite a bit of structure from the start.

IDL EX2 Nadav Eisen and Yonatan Microshnik

June 30, 2024

Question 1:

We'll prove that convolution is time/space invariant, where x marks the time/space variable in $f(x)$, and f_o marks the signal defined as $f_o(x) = f(o+x)$ the signal f with an offset o.

$$(g * f_o)(x) = \sum_y g(x-y) \cdot f_o(y)$$

$$= \sum_y g(x-y) \cdot f(y+o) = \sum_{y+z=x+o} g(y) \cdot f(z)$$

$$= \sum_y g((x+0)-y) \cdot f(y) = (g*f)(x+o)$$

i.e, a shift in the input causes the same shift in the output.

Therefore, by definition, convolution is time/space invariant.

Question 2:

a)

Time space invariant, as if we mark this layer L such that for any signal f we have $L[f](x) = f(x) + C$, we know of course that if we define an offset signal $f_t(x) = f(t+x)$, we get that:

$$L[f_t](x) = f_t(x) + C = f(t+x) + C = L[f](t+x)$$

i.e, a shift in the input causese the same shift in the output.

b)

Time space invariant, as if we mark this layer L such that for any signal f we have $L[f](x) = \begin{pmatrix} P_{RL}(f(x)_1) \\ \vdots \\ P_{RL}(f(x)_n) \end{pmatrix}$, where P_{RL} is any pointwise non-linearity function. We know that if we define an offset signal $f_t(x) = f(t+x)$, we get that:

$$L[f_t](x) = \begin{pmatrix} P_{RL}(f_t(x)_1) \\ \vdots \\ P_{RL}(f_t(x)_n) \end{pmatrix} = \begin{pmatrix} P_{RL}(f(t+x)_1) \\ \vdots \\ P_{RL}(f(t+x)_n) \end{pmatrix} = L[f](t+x)$$

i.e, a shift in the input causese the same shift in the output.

c)

Not time space invariant.

We can see by defintion that a layer L is time space invariant iff it commutes with the shift signal operator S_t, defined as $S_t[f](x) = f(x+t)$. However, for the signal vector $f = (1,2,3,4)$, strided max pooling with a factor of 2 which we'll mark L, and S_1, we can see that:

$$L\left[S_1\left[\begin{pmatrix}1\\2\\3\\4\end{pmatrix}\right]\right] = L\left[\begin{pmatrix}4\\1\\2\\3\end{pmatrix}\right] = \begin{pmatrix}4\\3\end{pmatrix}$$

And:

$$S_1\left[L\left[\begin{pmatrix}1\\2\\3\\4\end{pmatrix}\right]\right] = S_1\left[\begin{pmatrix}2\\4\end{pmatrix}\right] = \begin{pmatrix}4\\2\end{pmatrix}$$

Therfore, we know that $S_1[L[f]] \neq L[S_1[f]]$. Not only that, but no shift of $L[f]$ will make it equal to $L[S_1[f]]$.

d)

Not time space invariant.

We'll mark the shift operator S_t as before, the convolution layre C, the additive constant operator to be B, the ReLu layer to be R and the poooling layer to be P. Therefore, we saw in the previous section that $S_1[P[f]] \neq$

$P[S_1[f]]$. And since we saw that B, R, C are LTI, such that for any vector signal f and operator $L \in \{B, R, C\}$, we nkow that $S_1[L[f]] = P[L[f]]$. Therefore:

$$P[R[B[C[S_1[f]]]]] = P[R[B[S_1[C[f]]]]]$$

$$= P[R[S_1[B[C[f]]]]] = P[S_1[R[B[C[f]]]]]$$
$$\neq S_1[P[R[B[C[f]]]]]$$

Therefore, all in all these layers together do not commute with the shift operator, therefore the system is not LTI.

Question 3:

For all of these we'll look at the input signal of the layer as a vector $v \in \mathbb{R}^n$

a)

This layer can be described as the function $f(v) = v + C$, such that:

$$(Df)_v = Id_{n\times n}$$

b)

This layeer can be descirbed as the function $f(v) = Av$, where A is a matrix $A \in \mathbb{R}^{m\times n}$, such that:

$$(Df)_v = A_{m\times n}$$

c)

From what we saw in class, if we look at the input signal as a 1D vector $v \in \mathbb{R}^n$, then any convolution from the signal space $\mathbb{R}^n$ to $\mathbb{R}^n$ can be defined by a circulant matrix, such that every row of the matrix is the previous row shifted.

For any signal $u \in \mathbb{R}^n$, we can define the convolution $v * u$ as:

$$(u*v)_i = \sum_{k=0}^{n-1} u_{i-k \mod n} \cdot v_k = \left(\begin{pmatrix} u_n & u_{n-1} & \cdots & u_1 \\ u_1 & u_n & \cdots & u_2 \\ \vdots & & & \vdots \\ u_{n-1} & \cdots & u_1 & u_n \end{pmatrix} v \right)_i$$

Such that:

$$\frac{d}{dv}(u * v) = \frac{d}{dv}\left(\begin{pmatrix} u_n & u_{n-1} & \cdots & u_1 \\ u_1 & u_n & \cdots & u_2 \\ \vdots & & & \vdots \\ u_{n-1} & \cdots & u_1 & u_n \end{pmatrix} v\right) = \begin{pmatrix} u_n & u_{n-1} & \cdots & u_1 \\ u_1 & u_n & \cdots & u_2 \\ \vdots & & & \vdots \\ u_{n-1} & \cdots & u_1 & u_n \end{pmatrix}$$

Similarly in the general case,