CSC413H1 - Assignment 2

1.1.1: Each $\mathcal{B}$ is randomly sampled without replacement from the rows of X, so it lies in the row space of X. Next, since $\nabla_{\mathbf{w}_t}\mathcal{L}(\mathbf{x}_j, \mathbf{w}_t) = \nabla_{\mathbf{w}_t}\frac{1}{n}||\mathbf{w}_t^T\mathbf{x}_j - \mathbf{t}_j||_2^2 = \frac{2}{n}||\mathbf{w}_t^T\mathbf{x}_j - \mathbf{t}_j||_2^2\mathbf{x}_j$ is a scalar multiple of $\mathbf{x}_j$, every update is a linear combination of rows in X and thus stays within the span of X. This implies that the mini-batch SGD solution $\hat{\mathbf{w}}$ is in the span of X, and similar to in Assignment 1 we can write $\hat{\mathbf{w}} = X^T\mathbf{a}$ for some $\mathbf{a} \in \mathbb{R}^n$. Then, for any other solution $\mathbf{w}' \in \mathbb{R}^d$ where $X\mathbf{w}' = \mathbf{t}$, notice that

$$(\hat{\mathbf{w}} - \mathbf{w}')^T\hat{\mathbf{w}} = (\hat{\mathbf{w}} - \mathbf{w}')^T X^T\mathbf{a} = (X\hat{\mathbf{w}} - X\mathbf{w}')^T\mathbf{a} = (\mathbf{t} - \mathbf{t})^T\mathbf{a} = 0$$

so $(\hat{\mathbf{w}} - \mathbf{w}')$ is orthogonal to $\hat{\mathbf{w}}$. By the Pythagorean Theorem for vectors,

$$||\hat{\mathbf{w}} - \mathbf{w}'||_2^2 + ||\hat{\mathbf{w}}||_2^2 = ||\mathbf{w}' - \hat{\mathbf{w}}||_2^2 + ||\hat{\mathbf{w}}||_2^2 = ||\mathbf{w}'||_2^2 \geq ||\hat{\mathbf{w}}||_2^2,$$

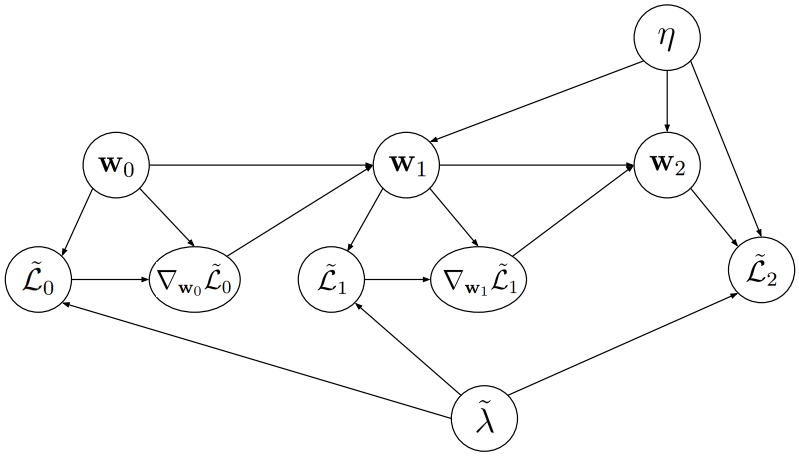
so $\hat{\mathbf{w}}$ is the minimum norm solution and $\hat{\mathbf{w}} = \mathbf{w}^*$.

1.2.1: RMSProp does not always obtain the minimum norm solution. Using the example in the hint, and setting $\eta = 0.5$, $\beta = 0.9$, and $\epsilon = 0.01$, we have

$$\begin{aligned}
\nabla_{\mathbf{w}_0}\mathcal{L}(\mathbf{w}_0) &= \frac{2}{n}||\mathbf{w}_0^T\mathbf{x}_1 - \mathbf{t}||_2^2\mathbf{x}_1 = 2(0-2)\begin{bmatrix}2\\1\end{bmatrix} = \begin{bmatrix}-8\\-4\end{bmatrix} \\
\mathbf{v}_0 &= \beta(\mathbf{v}_{-1}) + (1-\beta)(\nabla_{\mathbf{w}_0}\mathcal{L}(\mathbf{w}_0))^2 = 0.1\begin{bmatrix}(-8)^2\\(-4)^2\end{bmatrix} = \begin{bmatrix}6.4\\1.6\end{bmatrix} \\
\mathbf{w}_1 &= \mathbf{w}_0 - \begin{bmatrix}0.5(-8)/(\sqrt{6.4}+0.01)\\0.5(-4)/(\sqrt{1.6}+0.01)\end{bmatrix} \approx \begin{bmatrix}1.5749\\1.5687\end{bmatrix}.
\end{aligned}$$

Note that $\mathbf{w}_1$ is not in the span of $\mathbf{x}_1$, so it is not the minimum norm solution.

2.1.1:



2.1.2: The forward-propagation has a complexity of $O(1)$ and the back-propagation has a complexity of $O(t)$.

2.2.1: First, $\mathbf{w}_1 = \mathbf{w}_0 - \eta\nabla_{\mathbf{w}_0}\mathcal{L}_0 = \mathbf{w}_0 - \frac{2\eta}{n}X^T(X\mathbf{w}_0 - \mathbf{t}) = \mathbf{w}_0 - \frac{2\eta}{n}X^T\mathbf{a}$. Then,

$$\begin{aligned}\mathcal{L}_1 &= \frac{1}{n}||X\mathbf{w}_1 - \mathbf{t}||_2^2 = \frac{1}{n}||X(\mathbf{w}_0 - \frac{2\eta}{n}X^T\mathbf{a}) - \mathbf{t}||_2^2 = \frac{1}{n}||X\mathbf{w}_0 - \frac{2\eta}{n}XX^T\mathbf{a} - \mathbf{t}||_2^2 \\ &= \frac{1}{n}||\mathbf{a} - \frac{2\eta}{n}XX^T\mathbf{a}||_2^2 = \frac{1}{n}||(I - \frac{2\eta}{n}XX^T)\mathbf{a}||_2^2 = \frac{1}{n}\mathbf{a}^T(I - \frac{2\eta}{n}XX^T)^T(I - \frac{2\eta}{n}XX^T)\mathbf{a} \\ &= \frac{1}{n}\mathbf{a}^T(I - \frac{2\eta}{n}XX^T)^2\mathbf{a}.\end{aligned}$$

2.2.3: First, $\nabla_\eta\mathcal{L} = \frac{2}{n}\mathbf{a}^T(I - \frac{2\eta}{n}XX^T)(-\frac{2}{n}XX^T)\mathbf{a}$. Then,

$$\begin{aligned}\frac{2}{n}\mathbf{a}^T(I - \frac{2\eta}{n}XX^T)(-\frac{2}{n}XX^T)\mathbf{a} &= 0 \\ \mathbf{a}^T(I - \frac{2\eta}{n}XX^T)(XX^T)\mathbf{a} &= 0 \\ \mathbf{a}^T(XX^T - \frac{2\eta}{n}XX^TXX^T)\mathbf{a} &= 0 \\ \frac{2\eta}{n}\mathbf{a}^TXX^TXX^T\mathbf{a} &= \mathbf{a}^TXX^T\mathbf{a}\end{aligned}$$

and so

$$\eta = \frac{n}{2}\frac{\mathbf{a}^TXX^T\mathbf{a}}{\mathbf{a}^TXX^TXX^T\mathbf{a}} = \frac{n}{2}\frac{(X^T\mathbf{a})^T(X^T\mathbf{a})}{(XX^T\mathbf{a})^T(XX^T\mathbf{a})} = \frac{n}{2}\frac{(X^T\mathbf{a})^2}{(XX^T\mathbf{a})^2}.$$

2.3.1: $\nabla_{\mathbf{w}_0}\tilde{\mathcal{L}} = \nabla_{\mathbf{w}_0}(\frac{1}{n}||X\mathbf{w}_0 - \mathbf{t}||_2^2 + \tilde{\lambda}||\mathbf{w}_0||_2^2) = \frac{2}{n}X^T(X\mathbf{w}_0 - t) + 2\tilde{\lambda}\mathbf{w}_0$, so the expression using $\tilde{\mathcal{L}}$ is

$$\begin{aligned}\mathbf{w}_1 &= \mathbf{w}_0 - \eta(\frac{2}{n}X^T(X\mathbf{w}_0 - t) + 2\tilde{\lambda}\mathbf{w}_0) \\ &= (1 - 2\eta\tilde{\lambda})\mathbf{w}_0 - \frac{2\eta}{n}X^T(X\mathbf{w}_0 - t)\end{aligned}$$

Next, $\nabla_{\mathbf{w}_0}\mathcal{L} = \frac{2}{n}X^T(X\mathbf{w}_0 - \mathbf{t})$, so the expression using $\mathcal{L}$ and weight decay is

$$\mathbf{w}_1 = (1 - \lambda)\mathbf{w}_0 - \frac{2\eta}{n}X^T(X\mathbf{w}_0 - \mathbf{t}).$$

2.3.2: If the update steps are equivalent, we have

$$\begin{aligned}(1 - 2\eta\tilde{\lambda})\mathbf{w}_0 &= (1 - \lambda)\mathbf{w}_0 \\ 1 - 2\eta\tilde{\lambda} &= 1 - \lambda \\ \tilde{\lambda} &= \frac{\lambda}{2\eta}.\end{aligned}$$

3.1: The output is $\begin{bmatrix} 0 & 0 & -1 & -1 & -1 \\ -1 & -2 & 3 & 2 & 4 \\ 4 & 2 & 1 & 2 & -2 \\ -2 & 3 & 1 & 3 & -1 \\ 0 & -2 & 4 & -2 & 0 \end{bmatrix}$. The filter detects edges.

3.2: For the CNN architecture,

- Total number of neurons: $32^2 * 3 + 32^2 * 32 + 16^2 * 32 + 16^2 * 64 + 8^2 * 64 + 8^2 * 3 = 64704$
- Total number of parameters: $3^2 * 3 * 32 + 3^2 * 32 * 64 + 3^2 * 64 * 3 = 21024$.

For the FCNN architecture,

- Total number of neurons: $32^2 * 3 + 32^2 * 32 + 16^2 * 32 + 16^2 * 64 + 8^2 * 64 + 8^2 * 3 = 64704$
- Total number of parameters: $32^2 * 3 * 32 + 16^2 * 32 * 64 + 8^2 * 64 * 3 = 634880$.

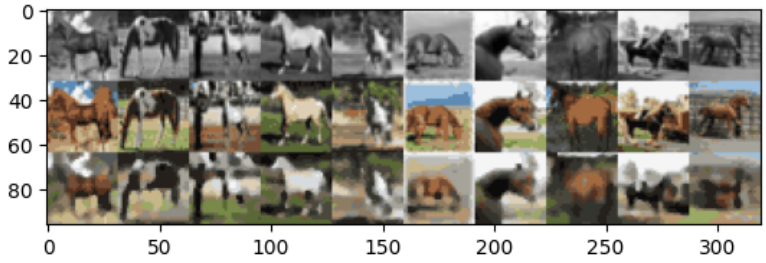
A disadvantage is that having more trainable parameters results in larger computational and memory complexities.

3.3: The receptive field can be affected by

- the number of pooling layers, since a pooling layer typically increases the field;
- the kernel size, since a larger kernel increases the field;
- and the depth of the neuron, since a neuron in a later layer can have a larger field than a neuron in an earlier layer.

4.1: Refer to the notebook.

4.2: The results are shown below. They look somewhat bad to me, since many pixels remained as greyscale when they should have been colored. Indeed, the final accuracy of 41.5% is quite low.



4.3: Denoting k as the kernel size,

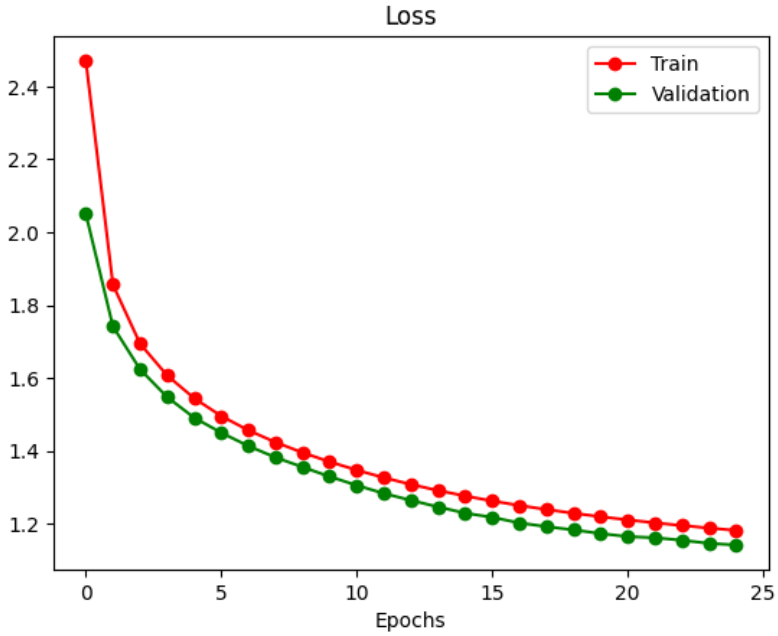
$$\begin{aligned}
\text{Total number of weights} &= k^2(NIC * NF + 2NF^2 + 2NF^2 + NF * NC + NC^2) \\
&= k^2(NIC * NF + 4NF^2 + NF * NC + NC^2) \\
\text{Total number of outputs} &= 32^2NF + 16^2NF + 16^2NF + 16^2(2NF) + 8^2(2NF) + 8^2(2NF) \\
&+ 8^2NF + 16^2NF + 16^2NF + 16^2NC + 32^2NC + 32^2NC + 32^2NC \\
&= 2880NF + 3328NC \\
\text{Total number of connections} &= 32^2k^2NIC * NF + 32^2NF + 16^2NF + 16^2k^2 * 2NF^2 + 16^2 * 2NF \\
&+ 8^2 * 2NF + 8^2k^2 * 2NF^2 + 16^2NF + 16^2NF + 16^2k^2NF * NC \\
&+ 32^2NC + 32^2NC + 32^2k^2NC^2 \\
&= 1024k^2NIC * NF + 640k^2NF^2 + 256k^2NF * NC \\
&+ 1024k^2NC^2 + 2432NF + 2048NC
\end{aligned}$$

If the input size is doubled, the total number of weights does not change, while the other two quantities increase by a factor of 4:

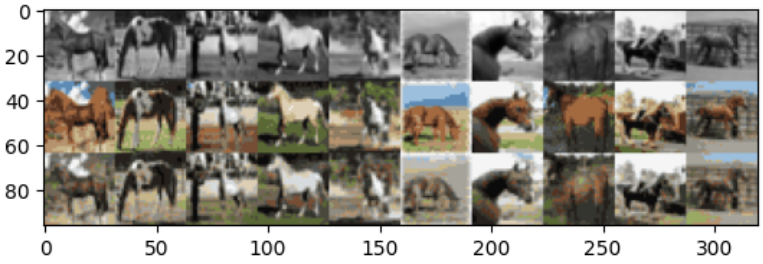
$$\begin{aligned}
\text{Total number of weights} &= k^2(NIC * NF + 4NF^2 + NF * NC + NC^2) \\
\text{Total number of outputs} &= 11520NF + 13312NC \\
\text{Total number of connections} &= 4096k^2NIC * NF + 2560k^2NF^2 + 1024k^2NF * NC \\
&+ 4096k^2NC^2 + 9728NF + 8192NC
\end{aligned}$$

5.1: Refer to the notebook.

5.2: The resulting plot is shown below:



5.3: The results are shown below; they look slightly better compared to before. At the end of training, the validation loss of the `ConvTransposeNet` model is lower than the loss of the `PoolUpsampleNet` model (1.1418 vs. 1.5791). This is likely because the max pool layers reduce the number of parameters and the upsampling layers do not increase them, while convolution transpose layers use more parameters and thus are able to learn more features.



5.4: For a kernel size of 4,

- the `padding` parameter for the `nn.Conv2d` layers should be 1;
- the `padding` parameter for the `nn.ConvTranpose2d` layers should be 1;
- and the `output_padding` parameter for the `nn.ConvTranpose2d` layers should be 0.

For a kernel size of 5,

- the `padding` parameter for the `nn.Conv2d` layers should be 2;
- the `padding` parameter for the `nn.ConvTranpose2d` layers should be 2;
- and the `output_padding` parameter for the `nn.ConvTranpose2d` layers should be 1.