Ross Altman
7/13/2018

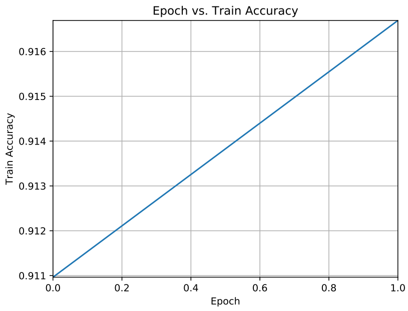
Lightmatter
Machine Learning Scientist

1 Report

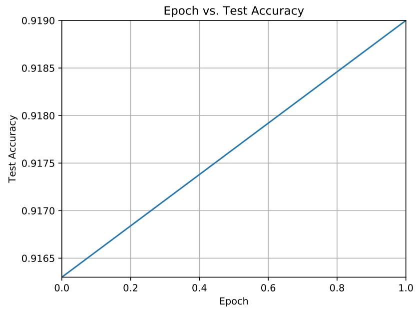
1.1 Part 1

I began by downloading TensorFlow's MNIST code for a simple neural network with one fully connected layer. After running the code as it was, I found an accuracy of 92% on the test set.

The results are shown in Figure ??.



(a) Part 1: Accuracy vs. Epoch as measured on the training set.



(b) Part 1: Accuracy vs. Epoch as measured on the test set.

1.2 Part 2

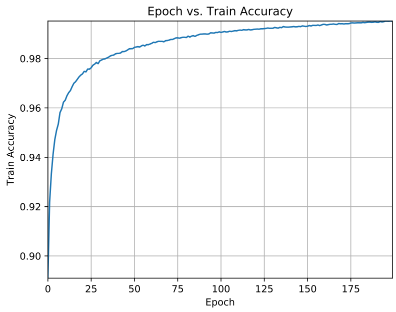
In the second part, I modified TensorFlow's MNIST code, as stated in the instructions, by expanding the neural network to include a total of 7 layers:

1. A 2d convolution layer with a 10×10 kernel, a stride of 1, 32 output channels, and a ReLU activation function.
2. A max pooling layer with a 2×2 kernel and a stride of 2.
3. Another 2d convolution layer with a 5×5 kernel, a stride of 1, 16 output channels, and a ReLU activation function.
4. Another max pooling layer with a 2×2 kernel and a stride of 2.
5. A flatten layer.
6. A fully connected layer with 1024 output channels and a ReLU activation function.

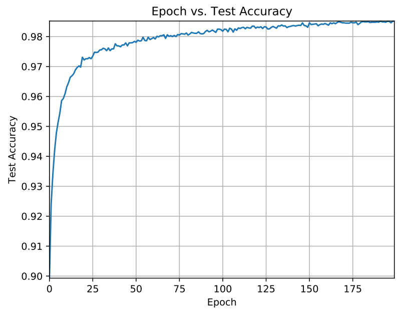
7. Another fully connected layer with 10 output channels and a Softmax activation function.

In addition, I used a cross-entropy loss function, the Adagrad optimizer with a learning rate of 10^-3. Each of the weight variables was initialized by a truncated normal distribution with mean 0 and standard deviation 0.1, and each bias variables was initialized to 0.1.

The results are shown in Figure 3.



(a) Part 2: Accuracy vs. Epoch as measured on the training set.



(b) Part 2: Accuracy vs. Epoch as measured on the test set.

1.3 Part 3

In the third part, I implemented the neural network from Part 2 using only `NumPy`. To do this, I set up the class structure shown in Figure 3.

In order to compute each of the forward and backward steps, I implemented the equations in Section 2. In order to compute the convolutional and max pooling layers faster, I first reshaped the order 4 tensor into a matrix via $x_{bijkl} \mapsto x_{(bIJ+iJ+j)(pQR+qR+r)}$ in order to use the full power of `NumPy`'s vectorization capabilities. After taking a dot product, I would reshape the matrix back into a tensor. And in order to speed up the reshaping, I implemented it via a Cython module.

Despite my attempts to speed up the calculations, this implementation was far too slow to run 200 epochs, so I ran 5. The trial did not result in convergence, and I have not had sufficient time to work out all the bugs, although I am confident that I can given enough time.

2 Mathematical Methods

2.1 General

2.1.1 Forward Propagation

$$\mathbf{x}^l = \sigma^{(l-1)}(\mathbf{z}^{(l-1)}) \tag{1}$$
$$\mathbf{z}^l = \mathbf{f}^l(\mathbf{x}^l, \mathbf{w}^l) + \mathbf{b}^l \tag{2}$$

2.1.2 Backpropagation

$$\begin{aligned}
\delta^l_{bi} = -\eta \frac{\partial J_b}{\partial z^l_{bi}} &= -\eta \frac{\partial J_b}{\partial x^L_{bj_0}} \frac{\partial x^L_{bj_0}}{\partial x^{(L-1)}_{bj_1}} \cdots \frac{\partial x^{(l+2)b}_{j_{L-l-2}}}{\partial x^{(l+1)b}_{j_{L-l-1}}} \frac{\partial x^{(l+1)b}_{j_{L-l-1}}}{\partial z^l_{bi}} \\
&= -\eta \frac{\partial J_b}{\partial x^L_{bj_0}} \left(\frac{\partial x^L_{bj_0}}{\partial z^{(L-1)}_{bk_0}} \frac{\partial z^{(L-1)}_{bk_0}}{\partial x^{(L-1)}_{bj_1}} \right) \cdots \left(\frac{\partial x^{(l+2)b}_{j_{L-l-2}}}{\partial z^{(l+1)b}_{k_{L-l-2}}} \frac{\partial z^{(l+1)b}_{k_{L-l-2}}}{\partial x^{(l+1)b}_{j_{L-l-1}}} \right) \frac{\partial x^{(l+1)b}_{j_{L-l-1}}}{\partial z^l_{bi}} \\
&= -\eta \frac{\partial J_b}{\partial x^L_{bj_0}} \frac{\partial \sigma^{(L-1)}_{bj_0}}{\partial z^{(L-1)}_{bk_0}} \frac{\partial f^{(L-1)}_{bk_0}}{\partial x^{(L-1)}_{bj_1}} \cdots \frac{\partial \sigma^{(l+1)b}_{j_{L-l-2}}}{\partial z^{(l+1)b}_{k_{L-l-2}}} \frac{\partial f^{(l+1)b}_{k_{L-l-2}}}{\partial x^{(l+1)b}_{j_{L-l-1}}} \frac{\partial \sigma^l_{bj_{L-l-1}}}{\partial z^l_{bi}}
\end{aligned} \tag{3}$$

$$\delta^{(L-1)}_{bi} = -\eta \frac{\partial J_b}{\partial x^L_{bj}} \frac{\partial \sigma^{(L-1)}_{bj}}{\partial z^{(L-1)}_{bi}} \tag{4}$$
$$\delta^{(l-1)}_{bi} = \delta^l_{bj} \frac{\partial f^l_{bj}}{\partial x^l_{bk}} \frac{\partial \sigma^{(l-1)}_{bk}}{\partial z^{(l-1)}_{bi}} \tag{5}$$

$$\Delta b^l_{bi} = -\eta \frac{\partial J_b}{\partial b^l_{bi}} = -\eta \frac{\partial J_b}{\partial z^l_{bi}} = \delta^l_{bi} \tag{6}$$
$$\Delta w^l_{bij} = -\eta \frac{\partial J_b}{\partial w^l_{bij}} = -\eta \frac{\partial J_b}{\partial z^l_{bk}} \frac{\partial f^l_k}{\partial w^l_{bij}} = \delta^l_{bk} \frac{\partial f^l_k}{\partial w^l_{bij}} \tag{7}$$

$$\Delta b^l_i = \frac{1}{B} \sum_{b=0}^{B} \Delta b^l_{bi} \tag{8}$$
$$\Delta w^l_{ij} = \frac{1}{B} \sum_{b=0}^{B} \Delta w^l_{bij} \tag{9}$$

2.2 Layers

1. Flatten:

$$F(x^l_{bijk}) = x^l_{b(iJK+jK+k)} \tag{10}$$

$$\frac{\partial F^l_{bu}}{\partial x^l_{cv}} = \delta_{bc}\,\delta_{uv}, \qquad \frac{\partial J}{\partial x^l_{c(iJK+jK+k)}} = (F^l_{cijk})^{-1}\left(\frac{\partial J}{\partial z^l_{c(iJK+jK+k)}}\right) \tag{11}$$

2. Fully Connected (Dense):

$$D^l_{bj}(\mathbf{x}^l, \mathbf{w}^l) = \sum_i x^l_{bi} w^l_{ij} \tag{12}$$

$$\frac{\partial D^l_{bj}}{\partial x^l_{ck}} = \delta_{bc}\, w^l_{kj}, \qquad \frac{\partial J}{\partial x^l_{ck}} = \frac{\partial J}{\partial z^l_{cj}} w^l_{kj} \tag{13}$$

$$\frac{\partial D^l_{bj}}{\partial w^l_{km}} = x^l_{bk}\,\delta_{jm}, \qquad \frac{\partial J}{\partial w^l_{km}} = \frac{\partial J}{\partial z^l_{bm}} x^l_{bk} \tag{14}$$

3. 2D Convolution:

$$C^l_{bijk}(\mathbf{x}^l, \mathbf{w}^l) = \sum_{p,q,r} x^l_{b(s_1 i+p)(s_2 j+q)r}\, w^l_{(P-p)(Q-q)rk} \tag{15}$$

$$\frac{\partial C^l_{bijk}}{\partial x^l_{ctuv}} = \delta_{bc}\, w^l_{(P+s_1 i-t)(Q+s_2 j-u)vk}, \qquad \frac{\partial J}{\partial x^l_{ctuv}} = \frac{\partial J}{\partial z^l_{cijk}} w^l_{(P+s_1 i-t)(Q+s_2 j-u)vk} \tag{16}$$

$$\frac{\partial C^l_{bijk}}{\partial w^l_{mnuv}} = x^l_{b(s_1 i+P-m)(s_2 j+Q-n)u}\,\delta_{kv}, \qquad \frac{\partial J}{\partial z^l_{bijv}} x^l_{b(s_1 i+P-m)(s_2 j+Q-n)u} \tag{17}$$

4. Max Pooling 2×2:

$$P^l_{bijk}(\mathbf{x}) = \max_{p,q} x^l_{b(s_1 i+p)(s_2 j+q)k} \tag{18}$$

$$\frac{\partial P^l_{bijk}}{\partial x^l_{ctuv}} = \begin{cases} \delta_{bc}\,\delta_{it}\,\delta_{ju}\,\delta_{kv} & x^l_{bijk} = P(x^l_{bijk}) \\ 0 & \text{otherwise} \end{cases}, \quad \frac{\partial J}{\partial x^l_{ctuv}} = \begin{cases} \frac{\partial J}{\partial z^l_{bijk}} & x^l_{bijk} = P(x^l_{bijk}) \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

2.3 Activation Functions

1. ReLU:

$$R^l_{bi}(\mathbf{z}^l) = \begin{cases} z^l_{bi} & z^l_{bi} \geq 0 \\ 0 & z^l_{bi} < 0 \end{cases} \tag{20}$$

$$\frac{\partial R^l_{bi}}{\partial z^l_{cj}} = \begin{cases} \delta_{bc}\, \delta_{ij} & z^l_{ci} > 0 \\ \\ \frac{1}{2}\delta_{bc}\, \delta_{ij} & z^l_{ci} = 0 \\ \\ 0 & z_{ci} < 0 \end{cases} \quad , \qquad \frac{\partial J}{\partial z^l_{cj}} = \begin{cases} \frac{\partial J}{\partial x^{l+1}_{cj}} & z^l_{cj} > 0 \\ \\ \frac{1}{2}\frac{\partial J}{\partial x^{l+1}_{cj}} & z^l_{cj} = 0 \\ \\ 0 & z^l_{cj} < 0 \end{cases} \tag{21}$$

2. Softmax:

$$S^l_{bi}(\mathbf{z}^l) = \frac{e^{z^l_{bi}}}{\sum\limits_k e^{z^l_{bk}}} \tag{22}$$

$$\frac{\partial S^l_{bi}}{\partial z^l_{cj}} = \begin{cases} \delta_{bc}\, S^l_{bj}(1 - S^l_{bj}) & i = j \\ \\ -\delta_{bc}\, S^l_{bi} S^l_{bj} & i \neq j \end{cases} \quad , \qquad \frac{\partial J}{\partial z^l_{cj}} = \left(\frac{\partial J}{\partial x^{l+1}_{cj}} - \sum_i \frac{\partial J}{\partial x^{l+1}_{ci}}\, S^l_{ci} \right) S^l_{cj} \tag{23}$$

2.4 Cost Function

Softmax cross-entropy:

$$J(\mathbf{z}^{(L-1)}) = \frac{1}{B} \sum_{b=0}^{B} J_b(\mathbf{z}^{(L-1)}) \tag{24}$$

$$J_b(\mathbf{z}^{(L-1)}) = \sum_i -y_{bi} \log(z^{(L-1)}_{bi}) \tag{25}$$

$$
\begin{aligned}
\frac{\partial J_b}{\partial z_{bj}^{(L-1)}} &= \frac{\partial J_b}{\partial x_{bk}^{L}} \frac{\partial S_k}{\partial z_{bj}^{(L-1)}} \\
&= -\frac{y_j}{x_{bj}^{L}} S_{bj}^{(L-1)}(1 - S_{bj}^{(L-1)}) + \sum_{k \neq j} \frac{y_k}{x_{bk}^{L}} S_{bk}^{(L-1)} S_{bj}^{(L-1)} \\
&= -\frac{y_j}{x_{bj}^{L}} x_{bj}^{L}(1 - x_{bj}^{L}) + \sum_{k \neq j} \frac{y_k}{x_{bk}^{L}} x_{bk}^{L} x_{bj}^{L} \\
&= -y_j(1 - x_{bj}^{L}) + \sum_{k \neq j} y_k x_{bj}^{L} \\
&= -y_j + x_{bj}^{L} \sum_{k} y_k \\
&= x_{bj}^{L} - y_j
\end{aligned} \tag{26}
$$

2.5 This Case

2.5.1 Forward Propagation

$$
L = 7 \tag{27}
$$

$$
\begin{aligned}
\sigma^0 &= \mathbf{R}(\mathbf{z}^0), & \mathbf{f}^0 &= \mathbf{C}(\mathbf{x}^0, \mathbf{w}^0) \\
\sigma^1 &= id(\mathbf{z}^1), & \mathbf{f}^1 &= \mathbf{P}(\mathbf{x}^1) \\
\sigma^2 &= \mathbf{R}(\mathbf{z}^2), & \mathbf{f}^2 &= \mathbf{C}(\mathbf{x}^2, \mathbf{w}^2) \\
\sigma^3 &= id(\mathbf{z}^3), & \mathbf{f}^3 &= \mathbf{P}(\mathbf{x}^3) \\
\sigma^4 &= id(\mathbf{z}^4), & \mathbf{f}^4 &= \mathbf{F}(\mathbf{x}^4) \\
\sigma^5 &= \mathbf{R}(\mathbf{z}^5), & \mathbf{f}^5 &= \mathbf{D}(\mathbf{x}^5, \mathbf{w}^5) \\
\sigma^6 &= \mathbf{S}(\mathbf{z}^6), & \mathbf{f}^6 &= \mathbf{D}(\mathbf{x}^6, \mathbf{w}^6)
\end{aligned} \tag{28}
$$

2.5.2 Backpropagation

$$
\begin{aligned}
\frac{\partial \sigma^6_{bi}}{\partial z^6_{bj}} &= \frac{\partial S_{bi}}{\partial z^6_{bj}}, & \frac{\partial f^6_{bi}}{\partial x^6_{bj}} &= \frac{\partial D_{bi}}{\partial x^6_{bj}}, & \frac{\partial f^6_{bi}}{\partial w^6_{bjk}} &= \frac{\partial D_{bi}}{\partial w^6_{bjk}} \\
\frac{\partial \sigma^5_{bi}}{\partial z^5_{bj}} &= \frac{\partial R_{bi}}{\partial z^5_{bj}}, & \frac{\partial f^5_{bi}}{\partial x^5_{bj}} &= \frac{\partial D_{bi}}{\partial x^5_{bj}}, & \frac{\partial f^5_{bi}}{\partial w^5_{bjk}} &= \frac{\partial D_{bi}}{\partial w^5_{bjk}} \\
\frac{\partial \sigma^4_{bi}}{\partial z^4_{bj}} &= \delta_{ij}, & \frac{\partial f^4_{bi}}{\partial x^4_{bj}} &= \frac{\partial F_{bi}}{\partial x^4_{bj}} \\
\frac{\partial \sigma^3_{bi}}{\partial z^3_{bj}} &= \delta_{ij}, & \frac{\partial f^3_{bi}}{\partial x^3_{bj}} &= \frac{\partial P_{bi}}{\partial x^3_{bj}} \\
\frac{\partial \sigma^2_{bpqr}}{\partial z^1} &= \frac{\partial R_{bijk}}{\partial z^2_{bpqr}}, & \frac{\partial f^2_{bijk}}{\partial x^2_{bpqr}} &= \frac{\partial C_{bijk}}{\partial x^2_{bpqr}}, & \frac{\partial f^2_{bijk}}{\partial w^2_{bpqrt}} &= \frac{\partial C_{bijk}}{\partial w^2_{bpqrt}} \\
\frac{\partial \sigma^1_{bi}}{\partial z^1_{bj}} &= \delta_{ij}, & \frac{\partial f^1_{bi}}{\partial x^1_{bj}} &= \frac{\partial P_{bi}}{\partial x^1_{bj}} \\
\frac{\partial \sigma^0_{bijk}}{\partial z^0_{bpqr}} &= \frac{\partial R_{bijk}}{\partial z^0_{bpqr}}, & \frac{\partial f^0_{bijk}}{\partial x^0_{bpqr}} &= \frac{\partial C_{bijk}}{\partial x^0_{bpqr}}, & \frac{\partial f^0_{bijk}}{\partial w^0_{bpqrt}} &= \frac{\partial C_{bijk}}{\partial w^0_{bpqrt}}
\end{aligned}
\tag{29}
$$

$$
\begin{aligned}
\Delta b^6_{bi} =& \delta^6_{bi} = -\eta \frac{\partial J_b}{\partial z^6_{bi}} & \Delta w^6_{bij} &= \delta^6_{bk} \frac{\partial D_{bk}}{\partial w^6_{bij}} \\
\Delta b^5_{bi} =& \delta^5_{bi} = \delta^6_{bj} \frac{\partial D_{bj}}{\partial x^6_{bk}} \frac{\partial R_{bk}}{\partial z^5_{bi}} & \Delta w^5_{bij} &= \delta^5_{bk} \frac{\partial D_{bk}}{\partial w^5_{bij}} \\
& \delta^4_{bi} = \delta^5_{bj} \frac{\partial D_{bj}}{\partial x^5_{bi}} \\
& \delta^3_{bijk} = \delta^4_{bp} \frac{\partial F_{bp}}{\partial x^4_{bijk}} \\
\Delta b^2_{bijk} =& \delta^2_{bijk} = \delta^3_{bpqr} \frac{\partial P_{bpqr}}{\partial x^3_{btuv}} \frac{\partial R_{btuv}}{\partial z^2_{bijk}} & \Delta w^2_{bijkm} &= \delta^2_{bpqr} \frac{\partial C_{bpqr}}{\partial w^2_{bijkm}} \\
& \delta^1_{bijk} = \delta^2_{bpqr} \frac{\partial C_{bpqr}}{\partial x^2_{bijk}} \\
\Delta b^0_{bijk} =& \delta^0_{bijk} = \delta^1_{bpqr} \frac{\partial P_{bpqr}}{\partial x^1_{btuv}} \frac{\partial R_{btuv}}{\partial z^0_{bijk}} & \Delta w^0_{bijkm} &= \delta^0_{bpqr} \frac{\partial C_{bpqr}}{\partial w^0_{bijkm}}
\end{aligned}
\tag{30}
$$

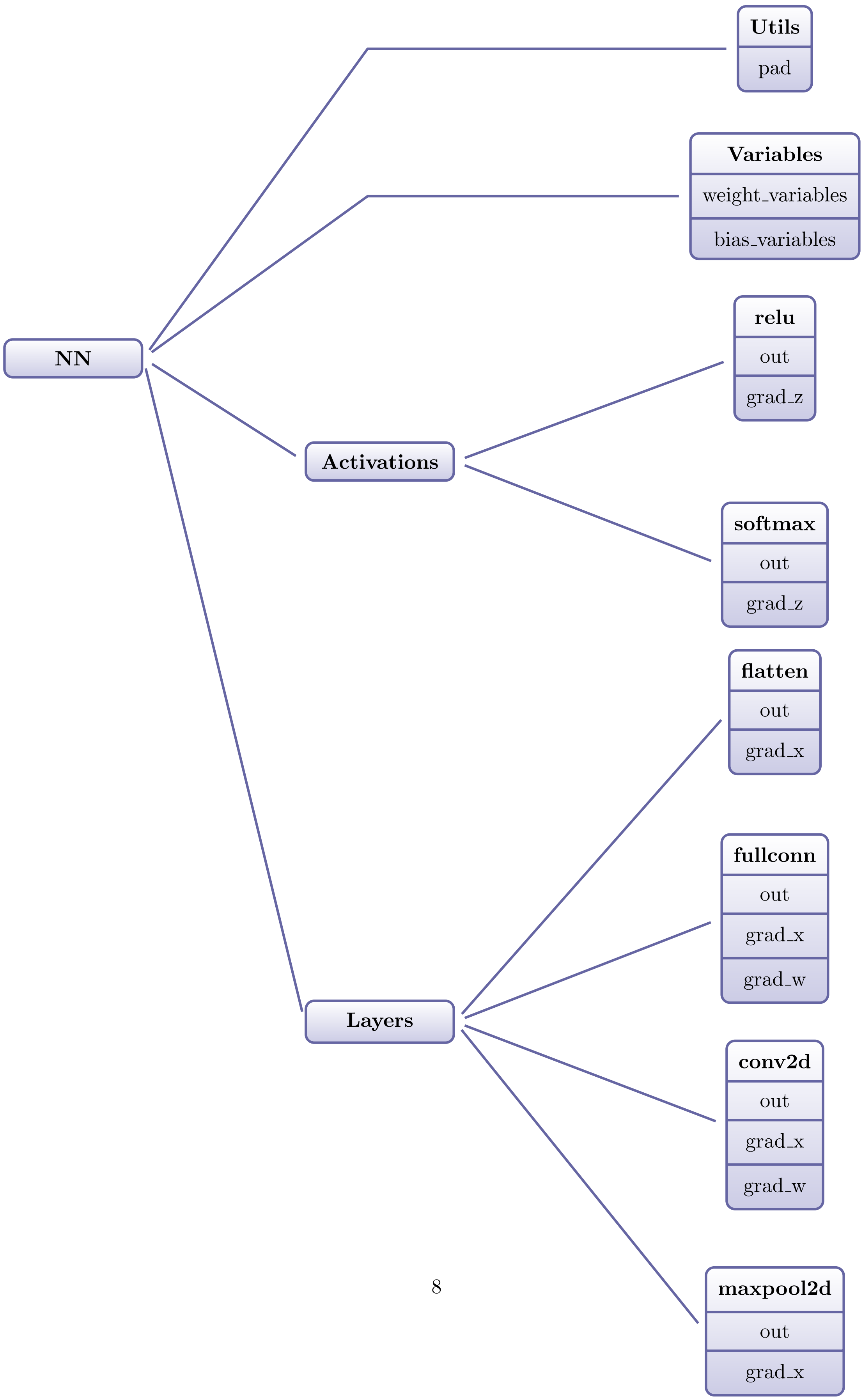


Figure 3: Class structure for neural network objects in Part 3.