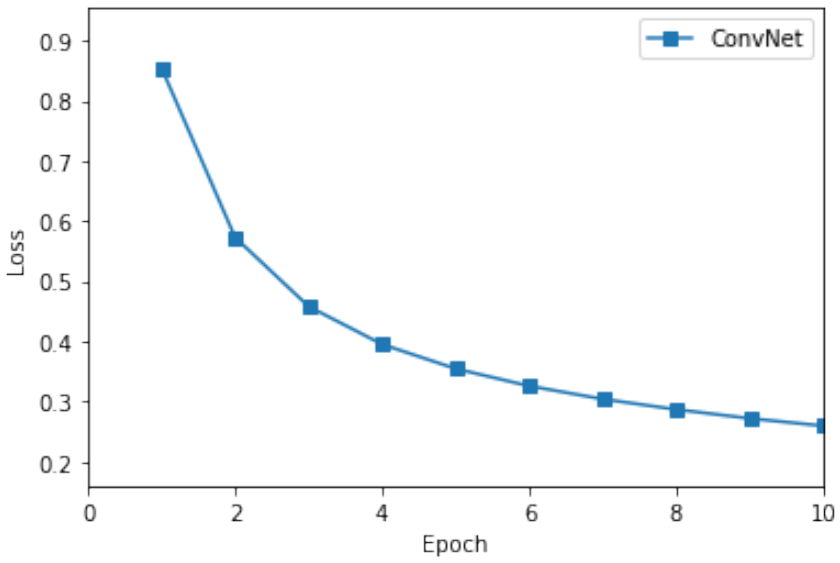
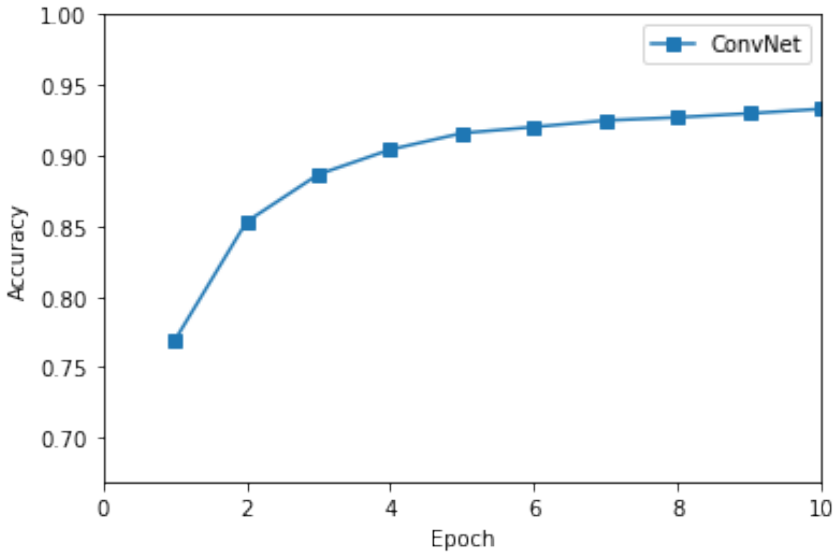
HW2 — MNIST Classification with CNN

1. CNN for MNIST Classification

A CNN was trained on the MNIST dataset with the following hyper-parameters:
- batch_size = 100
- max_epoch = 10
- learning_rate = 0.001
- weight_decay = 0.005

The final training and test performance is recorded below

	Training Data	Test Data
Loss	0.3299 	n/a
Accuracy	0.9123	0.9197
		

2. CNN vs MLP

Let us compare the performance between a 2-layer multilayer perceptron and the convolution neural network.

- The 2 layer MLP has the following architecture:
  - FC Layer —> ReLU —> FC Layer —> ReLU —> FC Layer —> Softmax Cross-Entropy
- The CNN has the following architecture:
  - Conv —> ReLU —> MaxPool —> Conv —> ReLU —> MaxPool —> Reshape —> FC Layer —> ReLU —> ***(Dropout)** —> FC Layer —> Softmax Cross-Entropy

The hyper-parameters used for both models are listed below:
- batch_size = 100
- max_epoch = 10
- init_std = 0.01
- learning_rate_SGD = 0.001
- weight_decay = 0.005 (0 for dropout network)
- dropout_rate = 0.2

2.1 MLP with Softmax Cross-Entropy and ReLU Activation

	Training	Validation	Test
Average Loss	0.7569	0.6797	n/a
Average Accuracy	0.8591	0.8988	0.8728

2.2 CNN with Softmax Cross-Entropy and ReLU Activation

	Training	Validation	Test
Average Loss	0.3299	0.2596	n/a
Average Accuracy	0.9123	0.9330	0.9197

2.3 CNN with Softmax Cross-Entropy and ReLU Activation and Dropout

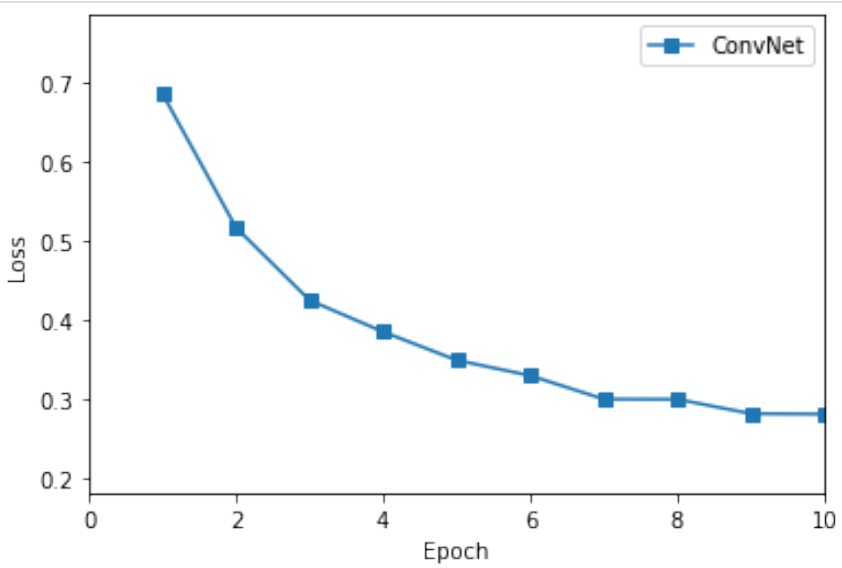
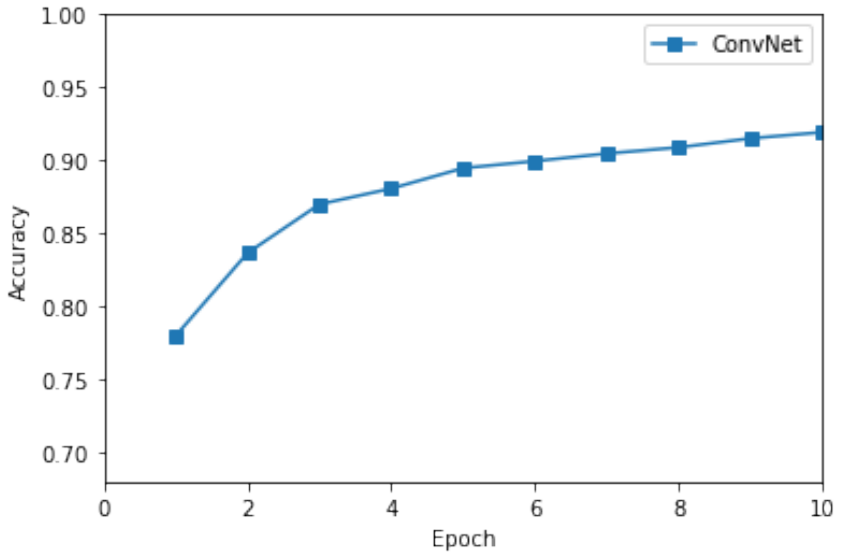
	Training	Validation	Test
Average Loss	0.3399	0.2813	n/a
Average Accuracy	0.8956	0.9192	**0.9305**

In terms of training time, the CNN with dropout took the longest to complete 10 epochs, followed by the CNN without dropout and then the MLP. This makes sense because the CNN with dropout has the most complexity of the 3 architectures.

In terms of convergence, after 10 epochs the MLP is the only architecture that seems to have converged. Both the CNN with and without dropout have not reached convergence. This is probably a result of both the increased complexity of these models and the increased learning power.

In terms of accuracy, both CNNs have all around higher accuracy than the MLP model. Although the regular CNN had higher training and validation accuracies, it had a lower test accuracy compared to the CNN with dropout. This is due to the generalization cause by the dropout layer.

3. Dropout

	Training Data	Test Data
Loss	0.3399	n/a
		
Accuracy	0.8956	0.9305
		

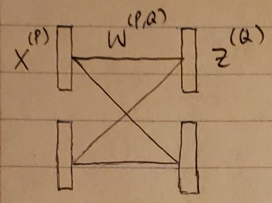
The dropout layer acts as a regularizer. I have tested multiple locations as well as different dropout rates and have observed the following:

- Dropout layers placed after Conv —> ReLU segments negatively affect classification performance. However, if the dropout rate is very small (<0.1) at these layers it results in comparable performance with no dropout.
- Dropout layers placed after Linear —> ReLU segments are the most effective at regularizing the model
- Dropout rates that are higher than 0.3 cause the model to underperform. This may be because CNN with dropout takes longer to converge, and thus must be trained over more epochs

4. Appendix: Mathematical Derivations

Batch CNN

BATCH CONVOLUTION LAYER



$$X \in \mathbb{R}^{B \times F_\ell \times H \times W}$$

$$Z \in \mathbb{R}^{B \times F_{\ell+1} \times (H - d_1 + 1) \times (W - d_2 + 1)}$$

$$W \in \mathbb{R}^{F_\ell \times F_{\ell+1} \times d_1 \times d_2}$$

B: batch size
F_ℓ: # feature maps at layer ℓ
H: height of feature map
W: width " "
d_1: height of filter
d_2: width of filter

FORWARD PASS

$$Z^{(N,Q)}_{m,n} = \sum_P \left(\sum_{i,j} X^{(N,P)}_{m+i,n+j} W^{(P,Q)}_{i,j} \right) + b^{(Q)}$$

$$= \sum_{P,i,j} X^{(N,P)}_{m+i,n+j} W^{(P,Q)}_{i,j} + b^{(Q)}$$

equal-valid correlation on P
valid correlation on i,j

$$\Rightarrow \quad Z^{(N,Q)} = X^{(N)} \circ_{valid} [W^{T(0,1)}]^{(Q)} + b^{(Q)}$$

* where $T(x,y)$ is the tensor transpose between axis x and axis y.

BACKWARD PASS

$$\frac{\partial L}{\partial X^{(N,P)}_{\ell,k}} = \sum_{Q,m,n} \frac{\partial L}{\partial Z^{(N,Q)}_{m,n}} \cdot \frac{\partial Z^{(N,Q)}_{m,n}}{\partial X^{(N,P)}_{\ell,k}}$$

$$= \sum_{Q,m,n} \delta^{(N,Q)}_{m,n} W^{(P,Q)}_{\ell-m,\,k-n}$$

equal-valid correlation on Q
full convolution on m,n

$$\Rightarrow \quad \frac{\partial L}{\partial X^{(N,P)}} = \delta^{(N)} \circ_{valid(0)} W^{(P)} \; *_{full(1,2)}$$

$$= \delta^{(N)} *_{\substack{valid(0) \\ full(1,2)}} flip_0 [W^{(P)}]$$

* where $flip_x$ denotes flipping the tensor along axis x.

The gradients of the weights and biases are...

$$\frac{\partial L}{\partial W^{(P,Q)}_{i,j}} = \sum_{N,m,n} \frac{\partial L}{\partial Z^{(N,Q)}_{m,n}} \cdot \frac{\partial Z^{(N,Q)}_{m,n}}{\partial W^{(P,Q)}_{i,j}}$$

$$= \sum_{N,m,n} \delta^{(N,Q)}_{m,n} \cdot X^{(N,P)}_{i+m,j+n}$$

equal-valid correlation on N
valid correlation on m,n

$$\Rightarrow \frac{\partial L}{\partial W^{(P,Q)}} = \left[X^{T(0,1)}\right]^{(P)} \circ_{valid} \left[\delta^{T(0,1)}\right]^{(Q)} \quad //$$

$$\frac{\partial L}{\partial b^{(Q)}} = \sum_{N,m,n} \frac{\partial L}{\partial Z^{(N,Q)}_{m,n}} \cdot \frac{\partial Z^{(N,Q)}_{m,n}}{\partial b^{(Q)}}$$

$$= \sum_{N,m,n} \delta^{(N,Q)}_{m,n} \cdot 1$$

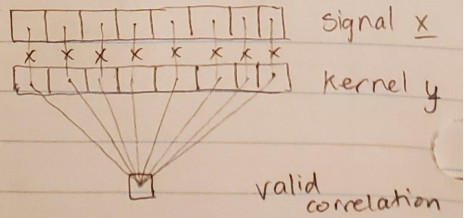
$$\Rightarrow \frac{\partial L}{\partial b^{(Q)}} = \sum_{N,m,n} \delta^{(N,Q)}_{m,n} \quad //$$

EQUAL-VALID CORRELATION

We can show that the dot product of 2 vectors is actually a form of valid correlation.

↳ $x \in \mathbb{R}^d$ } equal dimensions!
↳ $y \in \mathbb{R}^d$ }

$$\underline{x}^T \underline{y} = \sum_i x_i y_i = \underline{x} \circ_{valid} \underline{y} \quad //$$

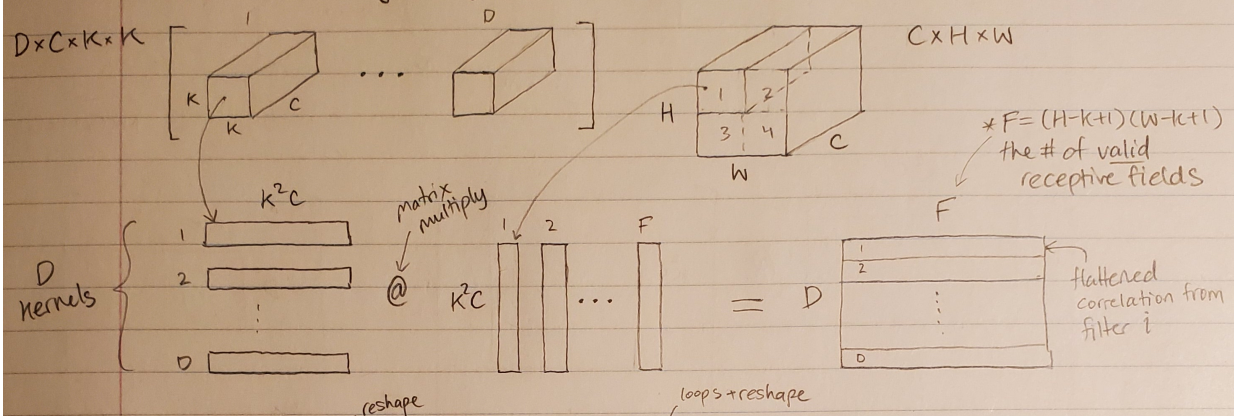


im2col

IM2COL

Im2col is a highly optimized routine used to turn valid convolution/ correlation into matrix multiplication.

↳ Let us consider the correlation between the feature maps of a single layer with D kernels/weights.



The resulting correlation tensor is obtained by reshaping the 2D matrix to the correct 3D tensor.

$$(D, F) \xrightarrow{\Delta} (D, H-K+1, W-K+1)$$

INDEXING ORDER

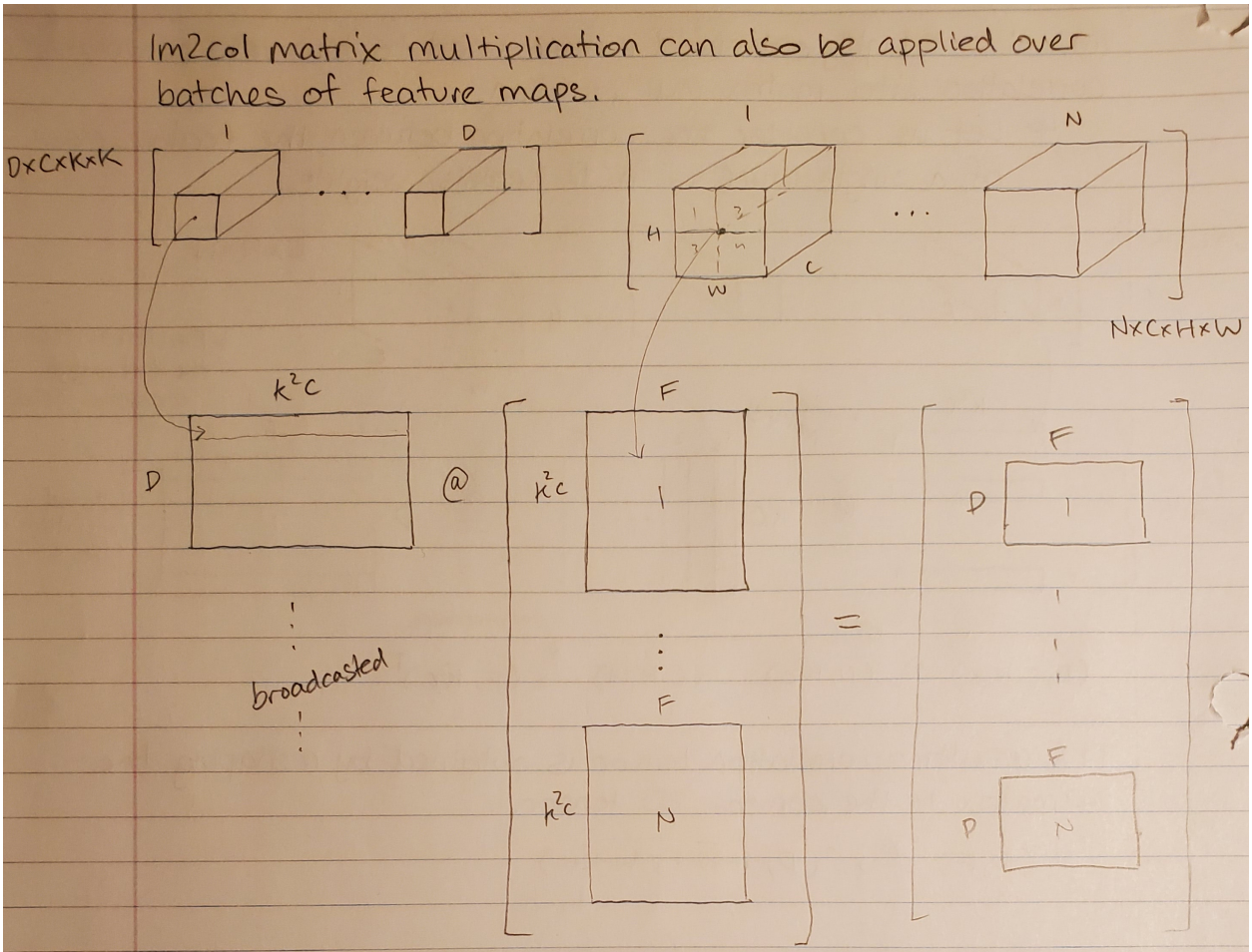
In order to execute im2col we must make sure maintain the same order when indexing, reshaping tensors.

↳ The order we implement is C-order

- In C-order the indexing is done from last axis to first axis.
  - ↳ for (C,H,W) dimensional tensor this means we index axis W, then H, then C.

```
for c in C:
    for h in H:
        for w in W:
            T[c,h,w]
```

- All reshaping should also use this order



The resulting tensor is obtained by reshaping...

$$(N, D, F) \xrightarrow{\Delta} (N, D, H-K+1, W-K+1)$$

- Therefore instead of applying 3D correlation in forward/backward pass we can apply im2col correlation

  ex.

$$Z^{(N,D)} = X^{(N)} \circ_{valid} W^{(D)}$$

$$\Rightarrow Z = X \circ_{im2col} W$$

$$X \in \mathbb{R}^{N,C,H,W}$$

$$W \in \mathbb{R}^{D,C,K,K}$$

$$Z \in \mathbb{R}^{N,D,H-K+1,W-K+1}$$