Homework 2
CNNs and Graphical Models[1]

CMU 10-417/617: Intermediate Deep Learning (Fall 2023)

https://rsalakhucmu.github.io/10417-23/
OUT: Wednesday, Oct 4, 2023

DUE: Wednesday, Oct 25, 2023
TAs: Jared Mejia, Kaiwen Geng

START HERE: Instructions

Homework 2 covers topics on convolutional neural networks and graphical models. The homework includes short answer questions, derivation questions, and a coding task.

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section on the course site for more information: https://rsalakhucmu.github.io/10417-23/#policies
- **Late Submission Policy:** See the late submission policy here: https://rsalakhucmu.github.io/10417-23/#policies
- **Submitting your work:**
  - **Written:** For both the programming portion and the written problems, we will be using Gradescope (https://gradescope.com/). For the programming portion, please submit your filled out "cnn.py" file to Gradescope under the assignment name "Homework 2 Programming", and please submit your writeup to "Homework 2 Written". Submissions for the written can be written in LaTeX or Word and must be submitted in a PDF form. **Handwritten solutions are accepted, but will receive zero credit if illegible.** Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. Each problem should be completed on a separate page. In addition, please tag the problems to the corresponding pages when submitting your

[1]Compiled on Friday 27th October, 2023 at 01:47

work. Tagging your pages will take at least 10 minutes, so please submit early enough that you do not miss the submission deadline. For more information about how to submit your assignment, see the following tutorial (note that even though the assignment in the tutorial is handwritten, submissions must be typed): https://www.youtube.com/watch?v=KMPoby5g_nE&feature=youtu.be

– **Code:** All code must be submitted to Gradescope. **If you do not submit your code to Gradescope, you will not receive any credit for your assignment.** Gradescope will be used to check for plagiarism. Please make sure your familiarize yourself with the academic integrity information for this course.

Problem 1 (12 Points)

In this problem, we are going to show that if we have an Boltzmann Machine, conditioning on a single variable results in a new Boltzmann Machine with updated potentials.

Suppose we have a graph G = G(V,E) with d nodes modeled using a Boltzmann Machine. We say the factorization of the model is defined as:

$$P_\theta(\mathbf{X}) = \frac{1}{Z(\theta)} \exp\left(\sum_{i,j} \theta_{i,j} x_i x_j + \sum_i \theta_i x_i\right)$$

We define the term $X_{-1} = \{V - X_1\}$ (all nodes except for X_1). Show that $P_\theta(X_{-1}|X_1 = 1)$ results in a Boltzmann Machine with edge weights J_{ij} and singleton values of $J_i + J_{1i} + J_{i1}$ with all remaining nodes (assuming X_1 is removed).

$$P_\theta(X) = \frac{1}{Z(\theta)} \exp\left[\sum_{i\neq 1, j\neq 1} \theta_{ij} x_i x_j + \sum_{i=1, j\neq 1} \theta_{ij} x_i x_j + \sum_{i\neq 1, j=1} \theta_{ij} x_i x_j + \sum_i \theta x_i\right]$$

$$= \frac{1}{Z(\theta)} \exp\left[\sum_{i\neq 1, j\neq 1} \theta_{ij} x_i x_j + \sum_{j\neq 1} \theta_{1,j} x_1 x_j + \sum_{i\neq 1} \theta_{i,1} x_i x_1 + \sum_i \theta x_i\right]$$

$$P_\theta(X_{-1} \mid x_1 = 1) = \frac{1}{Z(\theta)} \exp\left[\sum_{i\neq 1, j\neq 1} \theta_{ij} x_i x_j + \sum_{j\neq 1} \theta_{i,j} x_j + \sum_{i\neq 1} \theta_{ij} x_i + \sum_i \theta x_i\right]$$

$$= \frac{1}{Z(\theta)} \exp\left(\sum_{i\neq 1, j\neq 1} \theta_{i,j} x_i x_j + \left(\sum_{j\neq 1} \theta_{i,j} x_j + \sum_{i\neq 1} \theta_{i,j} x_i + \sum_i \theta x_i\right)\right)$$

Problem 2 (12 Points)

Consider he use of iterated conditional modes (ICM) to minimize the energy function (that we considered in class for image denoising example) given by:

$$E(\mathbf{x}, \mathbf{y}) = h \sum_i x_i - \beta \sum_{i,j} x_i x_j - \eta \sum_i x_i y_i, \tag{1}$$

where $x_i \in \{-1, 1\}$ is a binary variable denoting the state of pixel i in the unknown noise-free image, i and j are indices of neighboring pixels, and $y_i \in \{-1, 1\}$ denotes the corresponding value of pixel i in the observed noisy image. The joint distribution is defined as:

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{\mathcal{Z}} \exp\left(-E(\mathbf{x}, \mathbf{y})\right) \tag{2}$$

- (5 pts) Write down an expression for the difference in the values of the energy associated with the two states of a particular variable x_j, with all other variables held fixed, and show that it depends only on quantities that are local to x_j in the graph.
- (5 pts) Consider a particular case of the energy function above in which the coefficients $\beta = h = 0$. Show that the most probable configuration of the latent variables is given by $x_i = y_i$ for all i.

a) $E(x,y|x_j) = h\sum_{i\neq j} x_i - \beta \sum_{i\neq j}\sum_{k\neq j} x_i x_k - \eta \sum_{i\neq j} x_i y_i + h x_j - \beta x_j \cdot \sum_{k\neq j} x_k - \beta x_j^2 - \eta x_j y_j$

$E(x,y|x_j=1) - E(x,y|x_j=-1) = h\sum_{i\neq j} x_i - \beta \sum_{i\neq j}\sum_{k\neq j} x_i x_k - \eta \sum_{i\neq j} x_i y_i + h - \beta \sum_{k\neq j} x_k - \beta - \eta y_j$

$- \left(h\sum_{i\neq j} x_i - \beta \sum_{i\neq j}\sum_{k\neq j} x_i x_k - \eta \sum_{i\neq j} x_i y_i - h + \beta \sum_{k\neq j} x_k - \beta + \eta y_j \right)$

$= 2h - 2\beta \sum_{k\neq j} x_k - 2\eta y_j$

b) $\text{Max}(P(x,y)) \Rightarrow \text{Min}(E(x,y))$

Given $\beta = h = 0$, $E(x,y) = -\eta \sum_i x_i y_i$

$\text{Min}(E(x,y)) \Rightarrow \text{Max}\left(\sum_i x_i y_i\right)$

$x_i, y_i \in \{-1, 1\} \Rightarrow x_i y_i \in \{-1, 1\}$

$\max(x_i y_i) = 1 \Rightarrow x_i, y_i = -1$ (or) $x_i, y_i = 1$

$\Rightarrow x_i = y_i$

Problem 3 (12 Points)

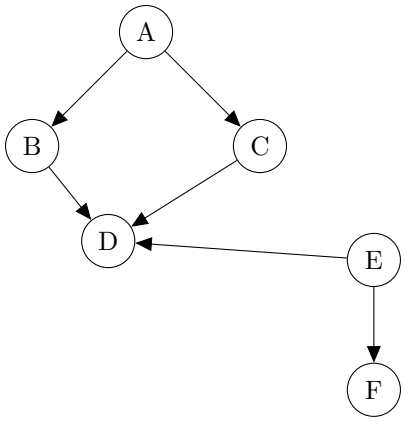


Figure 1: Directed Graphical Model

(Conditional) Independence. Answer the following questions with an explanation for why each statement is either true or false.

1. Is $A \perp\!\!\!\perp E$?

Yes, the connection between A and E is head-to-head, hence the two nodes are independent from each other as there no descendant (D) observed.

2. Is $A \perp\!\!\!\perp E \mid D$?

No, the variables A and E are not independent, as the value of descendant D has implications on the values of A and E.

3. Is $D \perp\!\!\!\perp F$?

No, D and F are not independent as the graph is tail-to-tail. Given information about the value of D we can learn something about the value of E which can then be used to learn something about the value of F

4. Is $D \perp\!\!\!\perp F \mid E$?

Yes, D and F are independent as the common cause E is observed.

Problem 4 (14 Points)

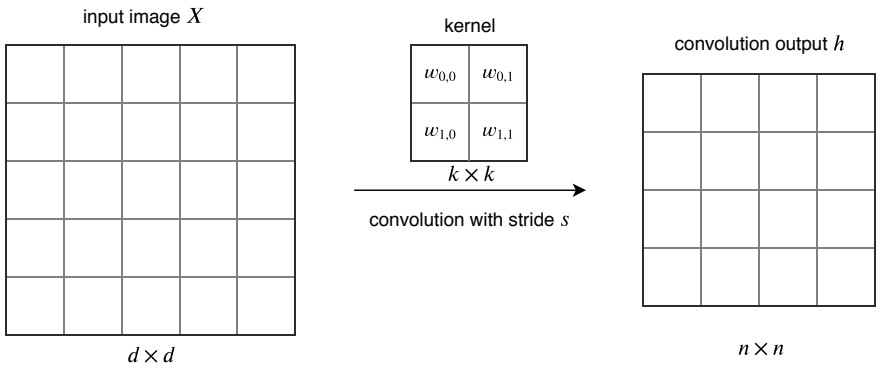


Figure 2: A simple example of convolution.

In this problem, you will work on derivations of convolution. You will also get familiar with `im2col` (image to column), a popular technique to accelerate convolution. For this problem, assume padding = 0.

Question 4.1 Let the input $X \in \mathbb{R}^{d \times d}$ be one single image, and is convolved by a $k \times k$ kernel with stride s. A simple example is shown in Figure 2.

a. (2 pts) The convolution output is in the shape of $n \times n$. Can you represent the output dimensionality n in terms of d, k and s?

n = (d - k)//s + 1

b. (3 pts) Denote the loss as L, convolution output as h, and kernel parameters as w. During backpropagation, given $\frac{\partial L}{\partial h_{i,j}}$ and input X, please derive $\frac{\partial L}{\partial w_{a,b}}$.

$$\frac{\partial L}{\partial w_{a,b}} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \frac{\partial L}{\partial h_{i,j}} \cdot \frac{\partial h_{i,j}}{\partial w_{a,b}} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \frac{\partial L}{\partial h_{i,j}} \cdot x_{si+a, sj+b}$$

c. (4 pts) Can the above derivative be represented as a convolution? If so, write down its expression and define the corresponding kernel.

$$\frac{\partial L}{\partial W_{a,b}} = \text{conv}\left(\frac{\partial L}{\partial h}, X\right)_{a,b}$$

The kernel is $\frac{\partial L}{\partial h}$, at i, j in the matrix the value of $\frac{\partial L}{\partial h}$ is $\frac{\partial L}{\partial h_{i,j}}$

Question 4.2 In this part, you will be walked through `im2col` (image to column), and you are encouraged to use this technique in the programming exercise. Naively, an simple implementation of convolution would be sliding the kernel over the feature map. However, this would result in loops of all kernels and all locations, which is slow and can be accelerated by constructing two big matrices and going through one single matrix multiplication.

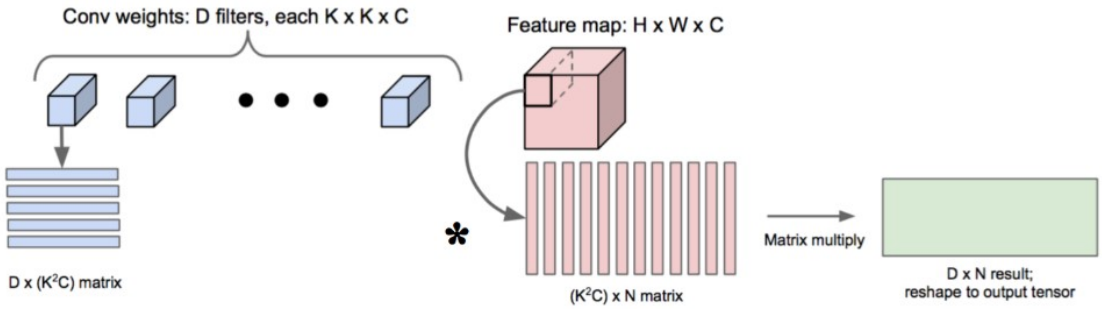


Figure 3: Illustration of im2col.

An illustration of im2col is shown in Figure 5. A patch inside the feature map corresponds to a possible location of the kernel. Each patch is of the same size as the convolution kernel, K^2C, and N is the number of all possible locations, or in other words, the spatial size of the output. We can take all possible patches, and reshape them into column vectors, then concatenate the vectors into one big matrix. Similarly, we can reshape the kernels into row vectors, and concatenate them into another one. Then we only need to go through one single matrix multiplication. And reshape the result to output shape.

For example, if the input feature map is of shape $32 \times 32 \times 3$, and we apply 5 filters each of size $5 \times 5 \times 3$ with stride=1 and padding=2, then the filter matrix would be of size 5×75, and image matrix would be of size 75×1024. The multiplication result would be of size 5×1024, and we need to reshape the result to shape $32 \times 32 \times 5$. (Hint: be very careful during reshaping.)

a. (3 pts) Consider a toy example where the input feature map is of size 3×3, and convolved by 2 kernels of size 2×2 with stride $s = 1$, as shown in Figure 4, please write the two constructed matrices with im2col.

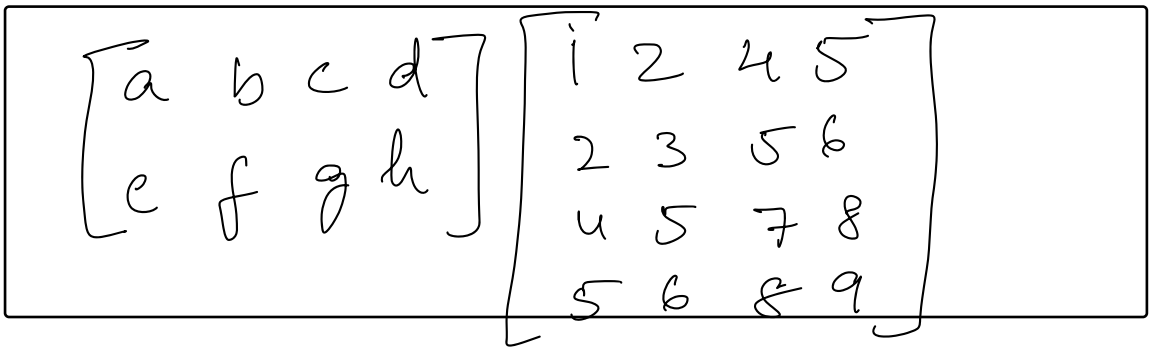
Feature map

1	2	3
4	5	6
7	8	9

Conv weights

a	b	e	f
c	d	g	h

Figure 4: A toy example of im2col



b. (2 pts) Give one significant drawback of im2col.

It creates multiple copies of the same data points leading to an increase in memory usage.

Problem 5 (10 Points) ***10-617 STUDENTS ONLY***

For an arbitrary graph consisting of nodes V, a *Markov blanket* of a node $v \in V$, denoted $\text{MB}(v)$, is any set of nodes (not containing v) that renders v conditionally independent of all other nodes in the graph. Formally,

$$\forall v \in V.\, p(v \mid V \setminus \{v\}) = p(v \mid \text{MB}(v)).$$

In a directed graph, the smallest such Markov blanket of a node v is the set of its parents, children, and co-parents (co-parents are nodes that share a child with v). Consider an (undirected) MRF $\mathcal{G} = (V, E)$ (V are vertices and E are edges)whose joint density can be factored into pairwise neighbor cliques,

$$p(V) = \exp\left\{ \sum_{(v_a, v_b) \in E} w_{ab} \Psi_{ab}(v_a, v_b) \right\},$$

where we have subsumed the normalizing constant into the weights w_{ab} such that the distribution integrates to 1.

Prove that for any node $v \in \mathcal{G}$, the set of v's neighbors is a Markov blanket for v. Pls don't just reference the slides.

Hint 1: Consider using Bayes Theorem

Hint 2: Definition of conditional probability

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Let N' denote $\mathbf{V} \setminus \{v\}$ (all nodes except v)

$$P(v \mid N') = \frac{P(v, N')}{P(N')}$$

$$P(v) = \exp\left(\sum_{v_a, v_b \in E} \left(w_{ab}\, \psi_{ab}(v_a, v_b) \right) \right)$$

$$P(N') = \sum_{\mathbf{V}} P(v) = \sum_{\mathbf{V}} \left[\exp\left(\sum_{v_a, v_b \neq v} w_{ab}\, \psi_{ab}(v_a, v_b) \right) \right]$$

$$P(v, N') = P(N', v) = P(N' \mid v) \cdot P(v)$$

$$P(v \mid N') = \frac{P(N' \mid v) \cdot P(v)}{P(N')}$$

Problem 6 (60 Points)

In this problem, we will implement convolutional neural networks for an image classification task. The data we use here is a subset of ImageNet dataset, where there are 10 classes of 32×32 colorful images.

Format of the data: A mapping from class names (string) to class ids (int) can be found in the `CLASS_IDS` variable within `cnn.py`. The data is in the format of a dictionary with both data and labels. We include starter code to load the Tiny ImageNet data and a helper function `prep_imagenet_data` to preprocess the numpy arrays for training. The `prep_imagenet_data` function outputs images in the form of a $(N, 3, 32, 32)$ numpy array of float64 scaled to [-1, 1] and labels in the form of a $(N, 10)$ one-hot encoded numpy array for both the training and the validation set.

Part 1, Implementation (28 pts)

In the programming folder, **cnn.py** is the only file you need to work on. **tests.py** and **tests.pk** are to help you test your implementation locally. Passing these local tests does not guarantee you passing the final online auto-grading tests, but failing locally very likely means also failing online. The following are terminal commands to run single module test and all modules test.

```
python -m unittest tests.TestReLU
python -m unittest tests
```

Below is a list of classes we will implement. Implement wherever the code template says `pass`. Reading `tests.py` might help you debug your implementation.

- **ReLU** (**0 points**): ReLU layer. Available unit tests: `TestReLU`.
- **Conv** (**14 points**): Convolutional layer. You will also need to implement functions **im2col** and **im2col_bw** in order to vectorize the forward and backward computation of convolutional layer [2]. Available unit tests: `TestIm2Col`, `TestIm2Col_bw`, `TestConvWeightsBias`, `TestConvForward`, `TestConvBackward` and `TestConvUpdate`.
- **Maxpool** (**6 points**): Max Pooling layer. Available unit tests: `TestMaxPoolForward` and `TestMaxPoolBackward`.
- **LinearLayer** (**4 points**): Fully Connected layer, or Linear Layer. Available unit tests: `TestFCWeightsBias`, `TestLinearForward`, `TestLinearBackward` and `TestLinearUpdate`.
- **SoftMaxCrossEntropyLoss** (**0 points**): The layer of softmax and cross-entropy loss. The input is the pre-softmax logits, and the output is the **sum** of cross-entropy loss across samples in a batch.

[2]The following links from the cs231n course at Stanford are very helpful: http://cs231n.stanford.edu/slides/2016/winter1516_lecture11.pdf(starting on slide 66) and http://cs231n.github.io/convolutional-networks/

- **ConvNet (4 points)**: This is where you will initialize and call all the previous layers to implement the convolutional network. Available unit tests: `TestConvNet`.

Tips:

- Linear layer weights initialization is the same with as in homework 1. Convolutional layer weights should be initialized as:

$$W_{i,j}^{k} \sim \texttt{Uniform}(-b, b) \ , \quad b = \sqrt{\frac{6}{(f+c) * h * w}}$$

  , where k, i, j are indices for network layer and nodes, f the number of filters (the number of output channels), c the number of input channels, h and w the filter height and width;

- You can start with these hyperparameters. Learning rate: 0.01, Momentum: 0.5, Batch size: 32. Other hyperparameters very likely could give better performance. Hyperparameters should be consistent when you compare different network architectures.

Im2col_helper Function

We have provided a helper file called **im2col_helper.pyc**. In this helper, we provide a binary that will help you test your solution code for the im2col and im2col_bw functions. Im2col and im2col_bw are non-trivial methods to implement, and can be difficult to implement efficiently.

Within your code, here is how you can use im2col_helper.py:

```
import im2col_helper

<Your Code>
...
shape = [20,10, 5, 1]
X =  np.random.choice(10, shape)
forward_output = im2col_helper.im2col(X=X, k_height=5, k_width=3, \
                    padding=1, stride=1)
backward_output = im2col_helper.im2col_bw(forward_output, X.shape, 5, 3)
...
<Your Code>
```

Feel free to use this helper function in other parts of your code; you will find that utilizing it in the Conv layer and MaxPool may speed up your experiments depending on your own implementation. However, if you call on the im2col_helper within your im2col implementation, you will lose 50 points on this assignment. If you choose to use the im2col_helper implementation you must uncomment the corresponding import line at the top of cnn.py and you must be using python version 3.10 (can use `conda create -n YOUR_ENV python=3.10`).

Part 2, Experiments (32 pts)

In this section, you will implement different network architectures to see how they affect performance.

The network models will take images as inputs and give softmax output over 10 classes. During training, we will perform gradient descent with momentum to minimize Cross-Entropy Loss.

1. **Data Exploration (5 pts)**. A critical aspect of training deep learning models is to understand the data you are working with. The goal of this section is to gain practice in doing so with visual data.

   (a) **Data Vis (2 pts)** Write a function that takes in the data split ('train' or 'val'), the number of samples to display per class ('num_samples_per_class'), and displays 'num_samples_per_class'. Use your function to generate a plot composed of two images per class on both the train set and the test set. Provide the resultant visualizations in the space below.

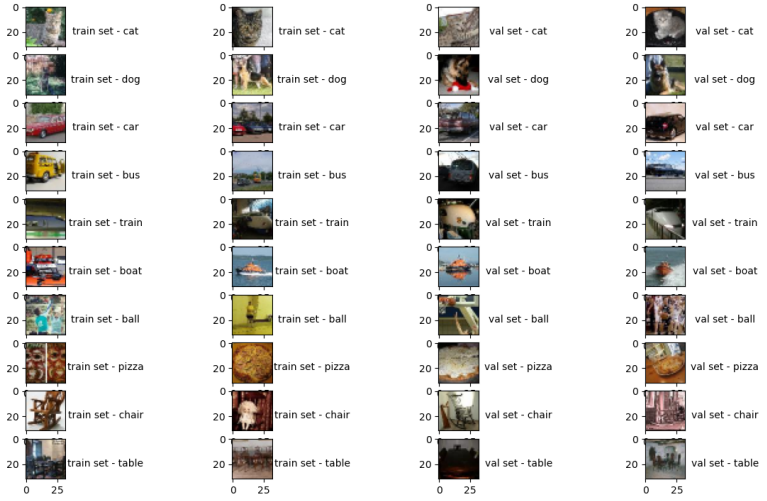


Figure 5: Samples of train and val data

(b) **Data Statistics (3 pts)** Write a function (or multiple functions) that takes in the data split ('train' or 'val') and outputs the number of samples per class, the data type, the data range (min and max), the data mean per image channel, and the data standard deviation per image channel. Provide your results for both the train and the test data. What do you notice?

data type : uint8

training data range: [0, 255]
validation data range: [0, 255]

All classes have 485 train examples and 48 validation examples

channel 0 training mean: 125.79
channel 0 validation mean: 125.04
channel 0 training standard deviation: 65.35
channel 0 validation standard deviation: 64.92
channel 1 training mean: 113.98
channel 1 validation mean: 113.34
channel 1 training standard deviation: 64.22
channel 1 validation standard deviation: 63.75
channel 2 training mean: 101.23
channel 2 validation mean: 101.02
channel 2 training standard deviation: 67.16
channel 2 validation standard deviation: 67.24

We observe that channel 2 has a significantly lower mean value and higher deviation compared to the channels 0 and 1.

2. **Training (12 pts)** For every architecture, plot the train and test loss together on one plot, with the loss on the y-axis against epoch number on x-axis. Similarly, plot the train and test accuracy after every epoch. Run the optimization for 50 epochs. Label each curve and all axes. Report the best loss and accuracy for training and testing achieved.

   For those with CNN layers, the stride is 1 and the padding size is 2.

   (a) **ConvNet (4 pts)** A network with a convolutional layer with 1 filter of size 5x5, then ReLU, then MaxPooling with a 2x2 filter and stride 2, then a linear layer with a softmax output.

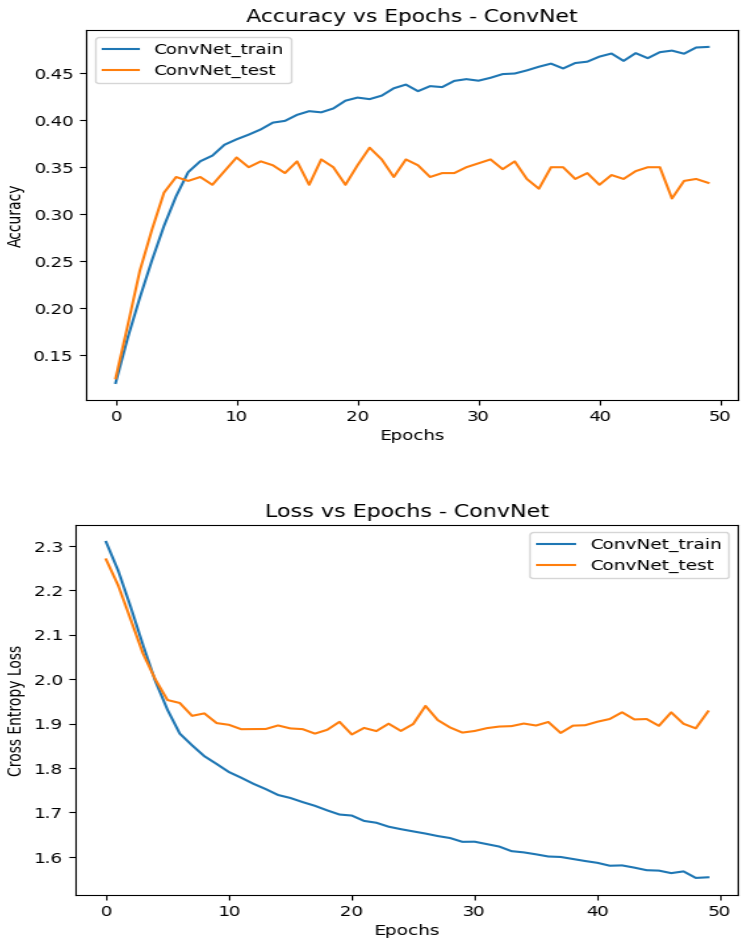


Figure 6: Loss and accuracy plots of *ConvNet*

Best Loss (Train, Test): (1.52, 1.89)
Best Accuracy (Train, Test) : (49.34, 37.48)

(b) **ConvNetThree (4 pts)** A convolutional layer with three blocks followed by a linear layer, a softmax layer. Each block should contain a convolutional layer with 16 filters of size 3x3, then ReLU, then MaxPooling with a 2x2 filter and stride 2, then Dropout with probability 0.1. Note that you may want to save a checkpoint of the model (i.e. with the pickle module), based on either the best performance or the last epoch, so you can evaluate it later.

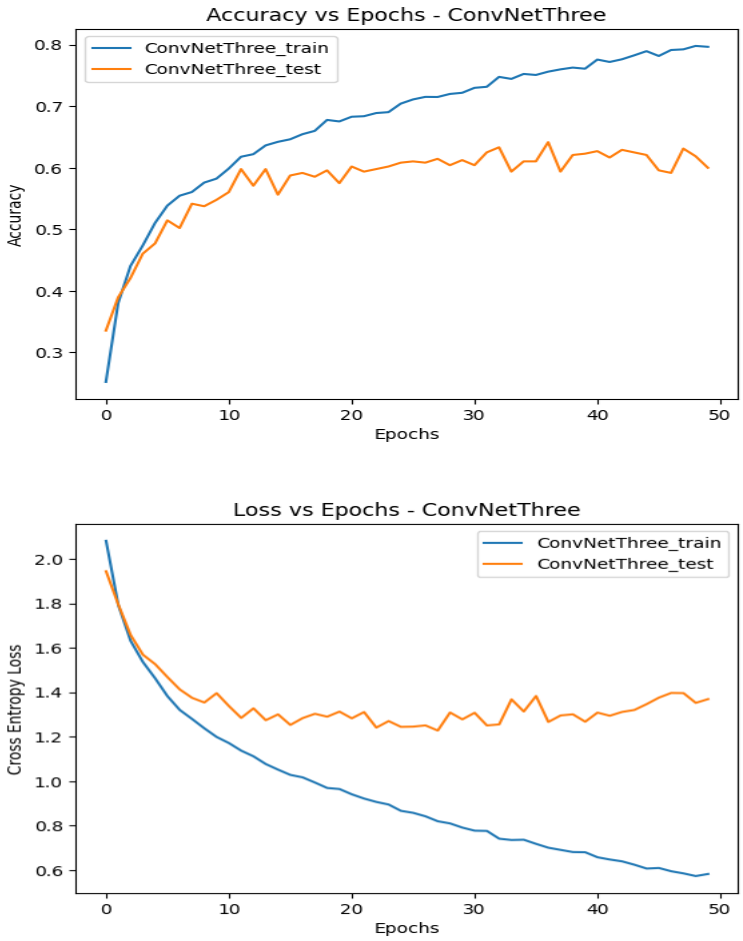


Figure 7: Loss and accuracy plots of *ConvNetThree*

Best Loss (Train, Test): (0.57, 1.23)
Best Accuracy (Train, Test) : (79.80, 63.17)

(c) **Model Tuning (4 pts)** Time to tune your CNN! Try to find the hyparameters and architecure that achieve the most optimal performance. For instance, you can experiment with different number of convolutional layers, drop-out rate, filter size, or even different activation functions. Report the highest performance (highest test accuracy) you get through your experiments and describe the architecture of your optimized model. If you prefer, you can visualize your architecture. You will be graded based on your reasoning for your design choices. Note that you may want to save a checkpoint of the model (i.e. with the pickle module), based on either the best performance or the last epoch, so you can evaluate it later.

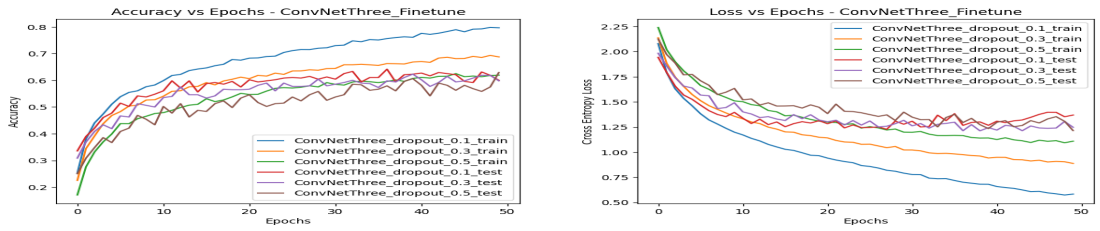


Figure 8: Loss and accuracy plots of *ConvNetThree*

To reduce the gap between train and test of ConvNetThree, we introduce regularization in the form of dropout. We try out three values of dropout 0.1, 0.3, 0.5. The model with 0.5 dropout results in a tighter train and test, but the model was still at a similar validation performance after 50 epochs as ConvNetThree. To improve the model I tried ConvNetThreeimproved wherein I increase the number of filters to $(64, 32, 16)$ in the three layer of ConvNetThree with dropout of 0.5. The motivation for the increasing the number of filters in the layers with a gradual decrease is because, the model needs to learn a large number of basic features hence 64 filters, as we pass to the next layer the filters the receptive field of the filters increase which translates to lesser number of filters to capture information from the previous layer.

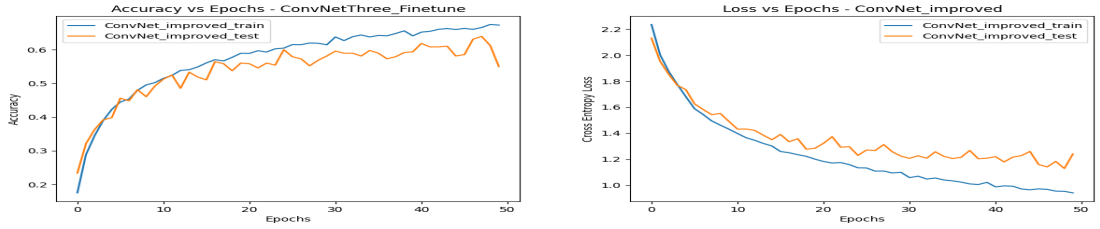


Figure 9: Loss and accuracy plots of *ConvNetThreeimproved*

Best Loss (Train, Test) - ConvThreeimproved : (0.94, **1.12**)
Best Loss (Train, Test) - ConvThree : (0.57, 1.23)
Best Accuracy (Train, Test) - ConvThreeimproved : (67.49, **64.73**)
Best Accuracy (Train, Test) - ConvThree : (79.80, 63.17)

3. **Evaluation (15 pts)** The goal of this next section is to gain practice understanding the performance and limitations of your trained models.

   (a) **(5 pts) Tiny ImageNet Evaluation.** Create a function to evaluate your model and plot a confusion matrix of the performance. Provide the results using both of the trained models from Question 2b and 2c on the TinyImageNet validation set. What do you notice? Include 5 misclassified images from each model, along with the predicted label and the ground truth label. Do the misclassified labels seem reasonable for these samples?

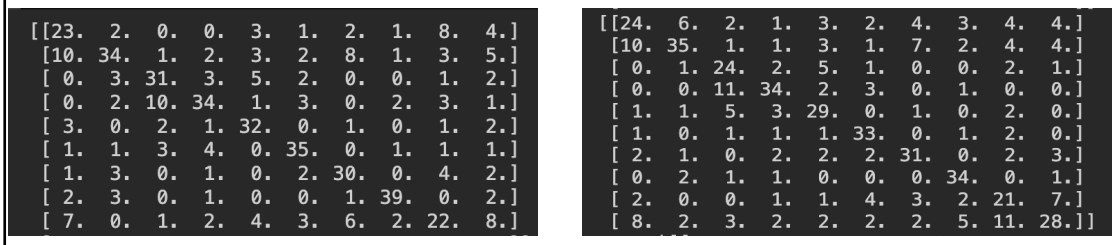


Figure 10: Confusion Matrix for ConvNetThree (Left) and ConvNetThreeimproved (Right)

Figure 11: Misclassified images using ConvNetThree (left) and ConvNetThreeimproved (Right)

Some of the misclassifications are reasonable as classifying a cat as a dog as because the whole body of the cat was not displayed or the face was at a slightly off angle. Some of the misclassifications however are unreasonable such as classifying a cat to table or a chair. There are no features similar between a chair/table and a cat.

(b) **(7 pts) COCO Evaluation.** So far, we've trained a model to predict images from the TinyImagenet dataset. Now, we'll pretend you're a Machine Learning Engineer for a new (imaginary) startup company, *DropoutAI*. Your boss wants to know how well your model performs IRL (*in the real world*). In order to convince him of your results, you decide to evaluate the model you designed (2c) against the model in prod (2b) on data from the COCO dataset (which can be found in the `10417-coco.pkl` file). Provide a confusion matrix of your results for both the model from 2b and 2c, and report the overall accuracy for each. Display five misclassified images for each model, what do the failure cases tell you about your models' predictions in the wild? How does performance for each model on COCO compare with the performance on Imagenet data?

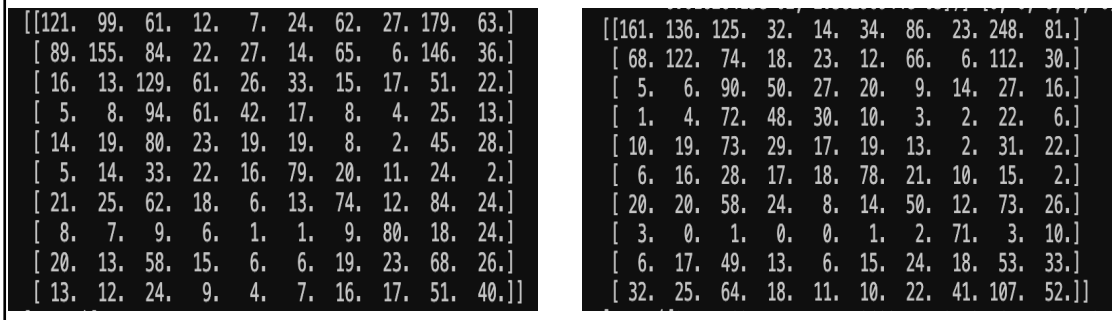


Figure 12: Confusion Matrix on COCO dataset ConvNetThree (left) (Acc:22.73) and ConvNetThreeimproved (Right) (Acc:23.88)

Figure 13: Misclassified images using COCO dataset ConvNetThree (left) and ConvNetThreeimproved (Right)

The misclassifications are similar to that of ImageNet, the cat labels are misclassified as dogs, tables and chair. The models perform worse compared to ImageNet, this is rather unexpected as the images are of the same labels that the model has been trained on. I would not be confident of my models performance in real life.

(c) **(3 pts) Reflection.** If you observed a difference in performance between parts (3a) and (3b), why do you think this is the case? (Hint: use your utilities from the Data Exploration Question). What will you tell your boss? Would you trust either of the two models in the wild? What metrics or evaluations would convince you that a computer vision model will be trustworthy in the wild?

The COCO dataset images are represented as floats whereas the ImageNet images are integers. While converting integers to floats there might be a difference in the resultant value which may lead to misclassification. Another difference which we observe is that the standard deviation in the COCO dataset is 0.25 across all channels, it is not the case in ImageNet. It is possible that the models are sensitive to the variance of the channels which may lead to poorer accuracies.
I would not trust either of my models to perform in the wild (real life examples).
Having a similar performance across various metrics (not just accuracy) on multiple diverse datasets would convince me that a computer vision model is trustworthy in real life.

Collaboration Questions Please answer the following:

After you have completed all other components of this assignment, report your answers to the collaboration policy questions detailed in the Academic Integrity Policies found here.

1. Did you receive any help whatsoever from anyone in solving this assignment? Is so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? Is so, include full details.
3. Did you find or come across code that implements any part of this assignment ? If so, include full details even if you have not used that portion of the code.

Solution

I found parts of the code to MaxPool here and im2col here