CSC420 – Assignment 2

Ansh Chokshi

1004996917

UtorID - chokshi7

17th February 2023

In solving the questions in this assignment, I worked together with my classmate Uttkarsh Berwal & 1005355018. I confirm that I have written the solutions/code/report in my own words.

As I have utilized it in my work, I would like to provide a reference to the given resource and tutorial - https://jovian.ml/ankitvashisht12/dog-breed-classifier-final#C55.

1)

Given I_0, the image of size $2^n \times 2^n$ and the Gaussian pyramid $\{I_0, I_1, I_2, \ldots I_{n-1}\}$ & the Laplacian pyramid $\{L_0, L_1, L_2 \ldots \ldots L_{n-1}\}$.

→ According to the definition of Gaussian Pyramid:

$$I_k = D\left(G_{blur}\left(I_{k-1}\right)\right)$$

- D → downsampling function
- G_{blur} → blurring function
- I_{k-1} → I_k is the Image at level k.

→ According to the definition of Laplacian Pyramid:

$$L_k = I_k - U(I_{k+1})$$

- L_k → K^{th} level of Laplacian Pyramid
- I_k → K^{th} level of guassian pyramid.

Above equation can be written as →

$$I_k = U(I_{k+1}) + L_k \quad \text{——} (1)$$

Lets take $K = 0$ for now.

$$I_0 = U(I_1) + L_0$$

→ Next we will use the kind of recursive process.

$$I_0 = U(U(I_2)) + L_1) + L_0$$
$$= U(U(U(I_3) + L_2) + L_1) + L_0$$
$$= U(U(U(I_3) + L_2)) + UL_1 + L_0$$
$$\vdots$$
$$= \prod_{i=1}^{n} U\, I_n + \sum_{i=0}^{n-1} \prod_{j=1}^{i} U\, L_i$$

Therefore, minimum information required is I_n which is of 1×1 size as the original image is $2^n \times 2^n$ that is downsampled by half n_{n-1} times.

$$= \prod_{i=1}^{n} U\, I_n + \sum_{i=0}^{n-1} \prod_{j=1}^{i} U\, L_i \longrightarrow \boxed{\text{Answer}}$$

2) Linear Model → $f(x, w) = wx + b$

* with limitation that data is not linearly separable

→ Now lets assume $\vec{x}$ be the input, $\vec{b}$ be the bias $\vec{w_i}$ be the weight.

→ Output is $f(x, w)$ and let it be $\vec{y_i}$, with i equal to i^{th} layer

→ We now have a fully connected neural network with linear activation function and n layers from $i = 1$ to $i = n$.

$\vec{y_1} = \vec{w_1}\ \vec{x} + \vec{b_1}$ [$\vec{y_1}$ becomes input for layer 2.]

$\vec{y_2} = \vec{w_2}\ \vec{y_1} + \vec{b_2}$

⋮

$\vec{y_n} = \vec{w_n}\ \vec{y_{n-1}} + \vec{b_n}$

→ Now implementing similar recursive approach as Question 1) part

$\vec{y_n} = \vec{w_n}\ (\vec{w_{n-1}}\ \vec{y_{n-2}} + \vec{b_{n-1}}) + \vec{b_n}$

$= \vec{w_n}\ (\vec{w_{n-1}}\ (\vec{w_{n-2}}\ \vec{y_{n-3}} + \vec{b_{n-2}}) + \vec{b_{n-1}}) + \vec{b_n}$

⋮

→ $\vec{y_n} = \vec{w_n} \cdot \vec{w_{n-1}} \cdots \vec{w_1}\ \vec{x} + \vec{w_n} \cdot \vec{w_{n-1}} \cdots \vec{w_2}\ \vec{b_1}$

$+ \vec{w_n} \cdot \vec{w_{n-1}} \cdots \vec{w_3}\ \vec{b_2} + \cdots \vec{w_n}\ \vec{b_{n-1}} + \vec{b_n}$

∴ $\vec{y} = \vec{w_{all}}\ \vec{x} + \vec{b_{all}}$ — (1) [here $w_{all} = \vec{w_n} \cdot \vec{w_{n-1}} \cdots \vec{w_1}$

[here $b_{all} = \vec{w_n} \cdot \vec{w_{n-1}} \cdots \vec{w_2}\ \vec{b_1} + \vec{w_n} \cdot \vec{w_{n-1}} \cdots \vec{w_3}\ \vec{b_2} + \cdots + \vec{w_n}\ \vec{b_{n-1}} + \vec{b_n}$]

From (1), we conclude that number of layers has effectively no impact on the network and we can express the output of such network as a function of its inputs. and its weights of layers.

3) a] Let us define.

$a_0 = w_1 * x_1$

$a_1 = w_2 * x_2$

$a_2 = w_3 * x_3$

$a_3 = w_4 * x_4$

$b_1 = a_0 + a_1$

$b_2 = a_2 + a_3$

$c_1 = -1 * b_1$

$c_2 = -1 * b_2$

$d_1 = e * p(c_1)$

$d_2 = e * p(c_2)$

$e_1 = d_1 + 1$

$e_2 = d_2 + 1$

$f_1 = e_1^{-1}$

$f_2 = e_2^{-1}$

$g_1 = f_1 \times w_5$

$g_2 = f_2 \times w_6$

$h_1 = g_1 + g_2$

$i_1 = -1 * h_1$

$j_1 = \exp(i_1)$

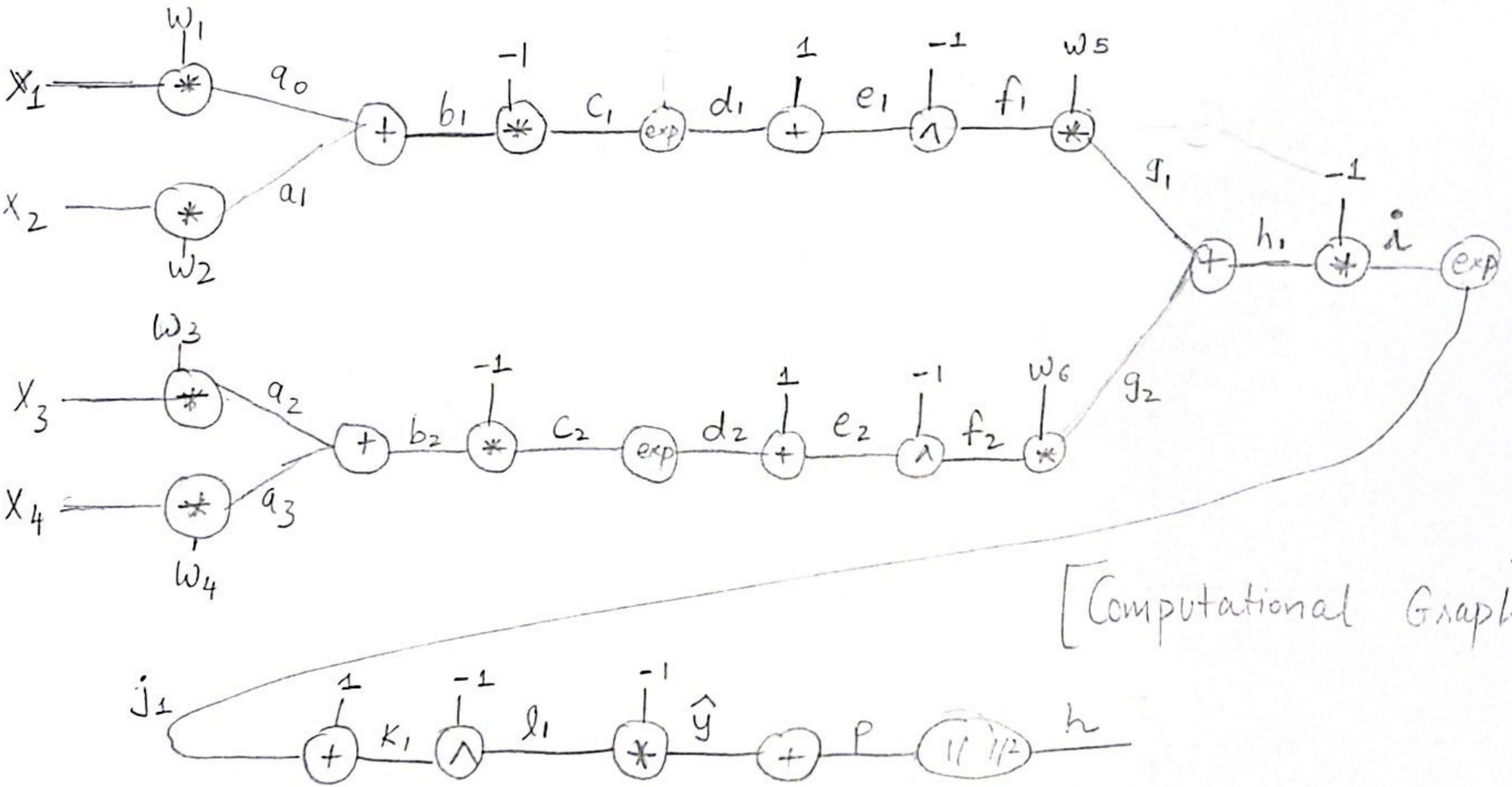
$k_1 = j_1 + 1$

$l_1 = k_1^{-1}$

$\hat{y} = l_1 * -1$

$p = y - \hat{y}$

$l = (y - \hat{y})^2$



[Computational Graph

3 b) Given $(x_1, x_2, x_3, x_4) = (1.2, -1.1, 0.8, 0.7)$

with true label, $y = 1.0$

$$\hat{y} = \sigma(w_5 \sigma(w_1 x_1 + w_2 x_2) + w_6 \sigma(w_3 x_3 + w_4 x_4))$$

$$L = (y - \hat{y})^2$$

$$O_1 = \sigma(w_1 x_1 + w_2 x_2)$$

$$O_2 = \sigma(w_3 x_3 + w_4 x_4)$$

Using Chain rull, we will find $\frac{\partial L}{\partial w_3}$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_2} \cdot \frac{\partial O_2}{\partial w_3}$$

$$\frac{\partial L}{\partial \hat{y}} = 2(y - \hat{y}) \quad \text{[given the hint]}$$

$$\frac{\partial \hat{y}}{\partial O_2} = w_6 \hat{y}(1 - \hat{y})$$

$$\frac{\partial O_2}{\partial w_3} = x_3 O_2 (1 - O_2)$$

$\hat{y} = 0.6801$ (substituting all the values)

$O_2 = 0.8749$

$y = 1, \quad w_6 = 0.93, \quad x_3 = 0.8$

$$\frac{\partial L}{\partial w_3} = 2(y - \hat{y}) \cdot w_6 \hat{y}(1 - \hat{y}) \cdot x_3 O_2 (1 - O_2)$$

$$= 2(1 - 0.6801) \cdot 0.93 \cdot 0.6801 \cdot (1 - 0.6801) \cdot 0.93 \cdot 0.8749 \cdot (1 - 0.8749)$$

$$= \boxed{0.0113}$$

4} Given Convolution layer
- input channels = 5
- Size of input channels = 12 x 12
- # of filters = 20
- Size of each filter = 4 x 4
- padding = 1
- Stride = 2

Layer P, pooling over each of C's output feature
3 x 3 local receptive fields and stride = 1

output height = (input height + padding top + padding bottom + (-kernel height)/2) + 1

$= \frac{(12 + 1 + 1 - 4)}{2} + 1 = \boxed{6}$ ← output height = output width

→ Padding layer, output = 6 − 3 + 1 = 4

→ Poding uses a max, so FLOPs = (3 x 3) − 1 = 8

→ FLOPs for pooling layer = 4 x 4 x 20 x 8 = 2560

→ For each and every entry, we have a kernel of size 4 x 4. Therefore the total will have 16 multiplications. that we used to multiply one element of kernel with one element of input matrix and then sum of that given result. We have 15 pairs to be added hence we have 15 additions

→ with bias: 20 x 6 x 6 x 50 x (4 x 4 x 2) + 2560 = 1154560 FlOPs

→ without bias: 20 x 6 x 6 x 50 x (4 x 4 x 2 − 1) + 1560 = 1118560 FlOPs

5) Here, the Trainable parameters is equal to the product of the kernel size and the number of input channels, and the number of output channels, plus the number of biases.

$C_1 = 5 \times 5 \times 1 \times 6 + 6 = 156$

$C_3 = 5 \times 5 \times 6 \times 16 + 16 = 2416$

$C_5 = 5 \times 5 \times 16 \times 120 + 120 = 48120$

→ The number of trainable parameters for the fully connected layers, is equal to the product of neurons of input layer and the output layer, plus the numbe of biases in output layer.

→ $F_6 = 120 \times 84 + 84 = 10164$

Output = $84 \times 10 + 10 = 850$

→ Here the pooling layers do not have the trainable parameters because their purpose is to downsample and extract, their operations are fixed and not adjustable during training.

Total trainable parameters = $C_1 + C_3 + C_5 + F_6 + \text{output}$

$= 156 + 2416 + 48120 + 10164 + 850$

$= \boxed{61706}$

6) Logistic Activation function :

$$f(x) = \frac{1}{1+e^{-x}} = \sigma(x)$$

Let us assume the output be :

$$y = \sigma(\underbrace{wx + b}_{\text{Let this be } z})$$

Now using the given hint →

$$\frac{dy}{dx} = \frac{dy}{dz} \cdot \frac{dz}{dx}$$

$$= \frac{e^{-z}}{(1+e^{-z})^2} \cdot \frac{dz}{dx}$$

$$= \frac{e^{-z}}{1+e^{-z}} \cdot \frac{1}{1+e^{-z}} \cdot w$$

$$= \left(1 - \frac{1}{1+e^{-z}}\right) \cdot \frac{1}{1+e^{-z}} \cdot w$$

$$= (1 - \sigma(z) \cdot \sigma(z) \cdot w)$$

$$= (1-y) \cdot y \cdot w$$

∴ We donot need input for the gradient, we can use just the output.

7) $\tanh(x) = \dfrac{1-e^{-2x}}{1+e^{-2x}}$

a) Ronge of tanh is $(-1,1)$ and for the logistic function it is $(0,1)$ tanh is shifted and scaled version of the logistic function

b) Gradient of $\tanh(x)$:

$$\frac{d\tanh(x)}{dx} = \frac{d}{dx}\left(\frac{1-e^{-2x}}{1+e^{-2x}}\right) = \frac{4e^{2x}}{(e^{2x}+1)^2}$$

$$= 4 \cdot \frac{e^{2x}}{e^{2x}+1} \cdot \frac{1}{e^{2x}+1}$$

$$= 4 \cdot \frac{e^{2x}}{e^{2x}+1} \cdot \frac{1}{e^{2x}+1}$$

$$= 4(1-6(2x)) \cdot 6(2x)$$

∴ The gradient of $\tanh(x)$ can be shown as a function of logistic function.

7 c) Sigmoid.

→ It is computationally faster

→ The rage is (0, 1) meaning we can use it for binary classification where output is a probability.

(2.) Tanh

→ Bigger / Larger gradient leads to faster convergence.

→ Range is (-1, 1) therefore we can use it for the prediction where sign matters.

Part 2 -

Task 1 – Inspection:

- The main difference I observed is that in DBI dataset the dogs are the main subject of the image and are in focus whereas in SDD dataset there are other objects or subjects in the images along with the dog. For example, a kid is there with his pug, a man with his dog, dog in a car, on the beach.
- This shows that SDD dataset images are more natural candid images whereas DBI dataset is like a passport photo of the dogs with good clear background and focus.
- DBI dataset has numerous white background pictures too. Basically, these are special photoshoot pictures of the dog breeds, whereas SDD are day to day pictures clicked by the dog owner or probably images sourced from the internet or social media.

Task 2 – simple CNN Training on the DBI:

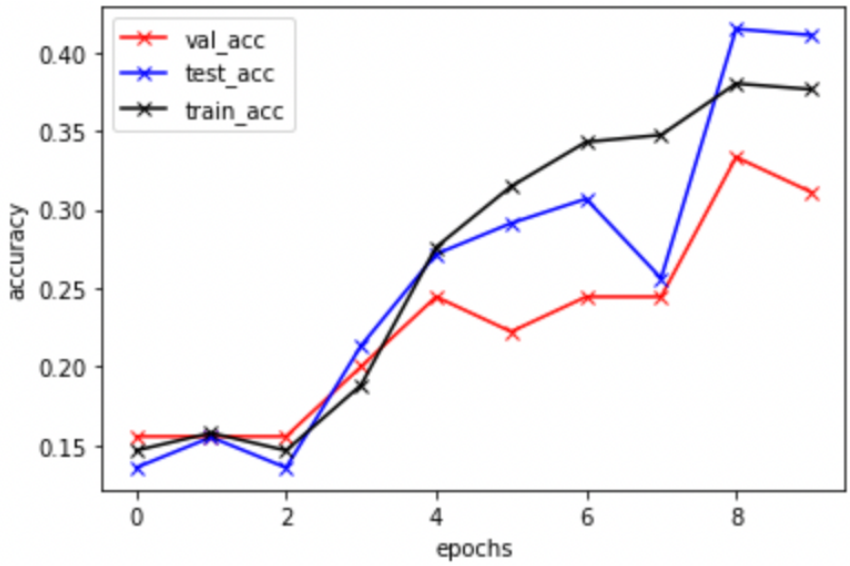


Figure 1 – CNN model with dropout

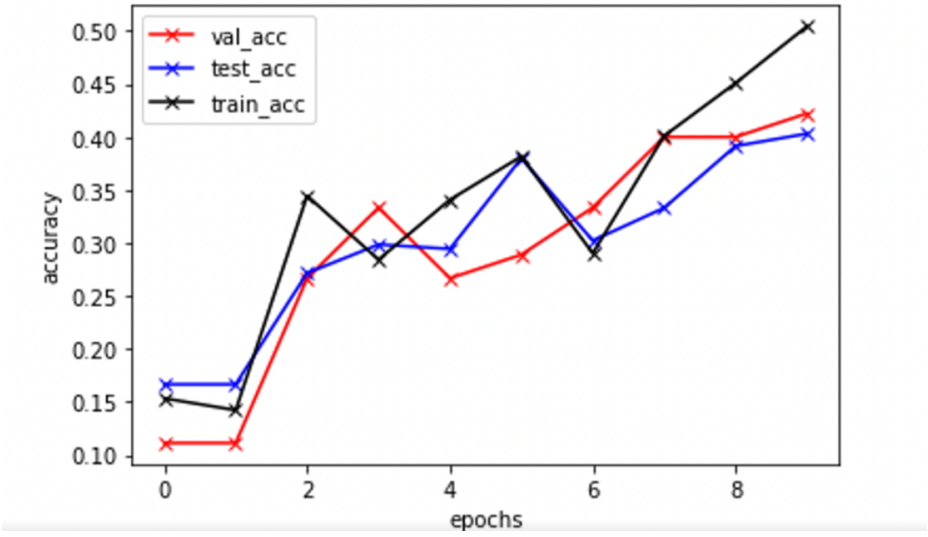


Figure 2– CNN model without dropout

Models:	CNN model + Dropout	CNN model w/o Dropout
DBI test dataset accuracy:	0.41968911917098445	0.388756471683921

In this scenario, the use of dropout leads to a moderate improvement in generalization, resulting in a slightly higher accuracy compared to the model without dropout. Towards the end of the training process, the test accuracy of the model with dropout catches up with its training accuracy, while in the model without dropout, the test accuracy remains noticeably lower than the training accuracy. Although dropout is effective in preventing overfitting, its benefits are less pronounced when the size of the training dataset is small. Overall, due to its slight improvement in generalization, the model with dropout has slightly accuracy than the one without.

Task 3 – ResNet Training on the DBI:

3 a).

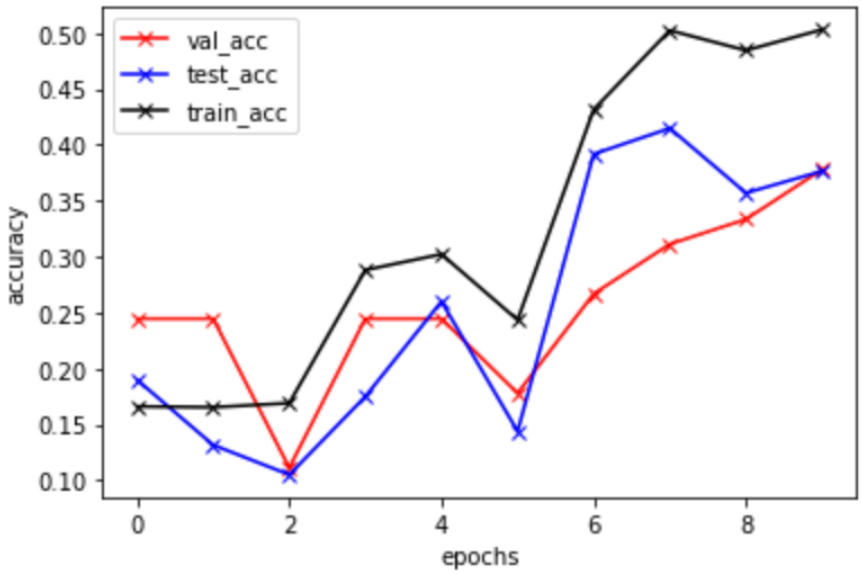


Figure 3– ResNet18 Untrained with DBI dataset

3 b).

Test dataset:	DBI test dataset	Entire SDD dataset
ResNet18 Untrained model accuracy:	0.39378238341968913	0.28872668288726683

- The accuracy is higher on DBI test dataset.
- The model's generalization capability is limited due to being trained on the DBI dataset, which only includes images of dogs with a clear background in focus. In contrast, the SDD dataset is much larger, includes multiple subjects, and has less emphasis on dogs. Consequently, the model may perform poorly on the SDD dataset since it has not been trained on such a diverse range of images. This lack of generalization results in a weak ability to predict images outside its trained scope.

Task 4– Fine-tuning on the DBI:

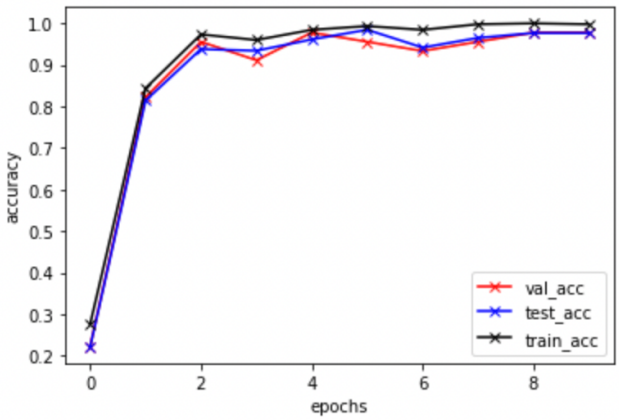


Figure 4 – ResNet18 Pretrained

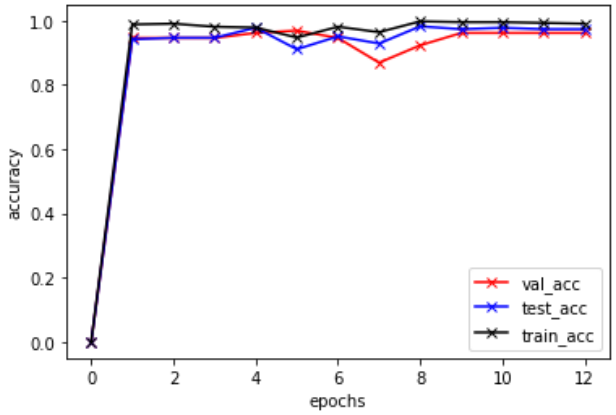


Figure 5 – ResneXt32 Pretrained

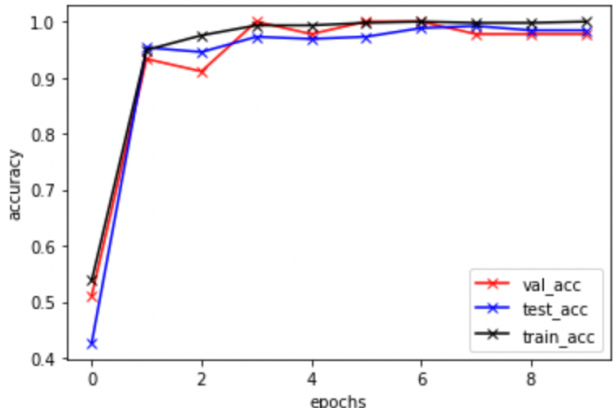


Figure 6 – ResNet34 Pretrained

Pertained models:	ResNet18	ResNeXt32	ResNet34
DBI test dataset:	0.9740932642487047	0.9844559585492227	0.9844559585492227
SDD entire dataset:	0.894566098945661	0.9261962692619627	0.9213300892133008

- The test accuracy on DBI test dataset is same across all three models.
- The test accuracy on SDD entire dataset with ResNet18 model is slightly (~3%) lower. Whereas the accuracy of the other two models is equal, could be result of larger size and more computes.
- The loss of training is very low and test accuracy very high, clearly in my opinion all three models are overfitting when observed the confusion matrix and the plots.

Task 5 –

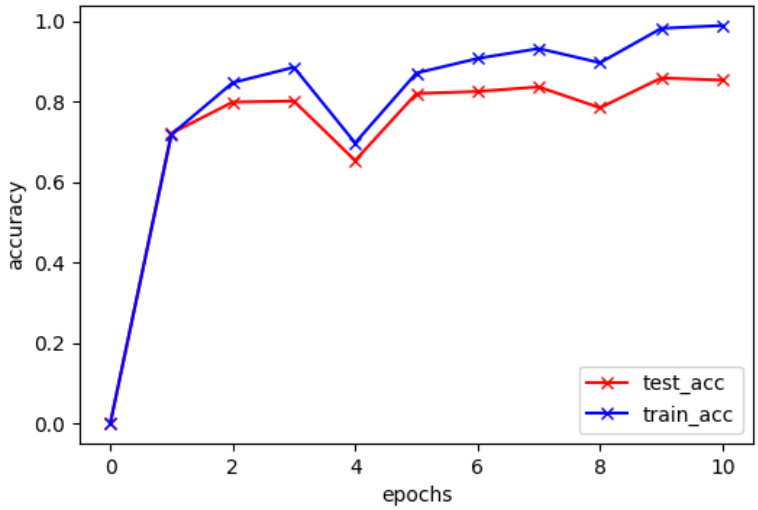


Figure 7 – ResNet18 Pretrained Dataset Classification

Accuracy of Resnet18 on dataset classification: 0.8535286284953395

- I wrote a manual script on my macOS to distribute the images in two folder SDD and DBI according to their origin. I decided to do this because it would fit directly with my Data setup code of pervious tasks.
- I decided to train the ResNet 18 Pretrained model (max_lr = 0.01, grad_clip = 0.1, weight_decay = 1e-4) along with SGD optimizer.
- ResNet18 is efficient small model that performs accurately with approx. 2000 images binary classification. Since it is a pretrained model, I can leverage the transfer learning to improve the accuracy. It is also ideal when working with a small dataset because it has already been trained on a large dataset and can be effective.
- The model achieves 85% accuracy and by observing the plot and confusion matrix we can conclude that it is not overfitting and efficient.

Task 6 –

Q. At training time, we have access to the entire DBI dataset, but none of the SDD dataset. All we know is a high-level description of SDD and its differences with DBI (similar to the answer you provided for Task I of this question).

- We have a dataset with an ideal setting with a focus on the subject in this case. We aim to train a model with the above dataset and make it accurate. To do so, we will have to do random transformation, cropping, blurring, cropping, zoom in/out, add noise, and filters, and change the background to make the images as non-ideal as possible. Not the whole dataset, we can add more non-ideal setting images in the current dataset or just transform x% of the images. This will help train our dataset in non-ideal settings such as the SDD dataset, improving the generalization of our model to predict images outside the scope of the DBI dataset.

Q. At training time, we have access to the entire DBI dataset and a small portion (e.g. 10%) of the SDD dataset.

- In this case, we will leverage the 10% SDD dataset which is the non-ideal setting of the images and subject (dog in our case). We will use that 10% of images to transform at least 20-30% of DBI images using the same type of noise, blur, backgrounds as of the 10% SDD dataset. Again, training the dataset with a mix of ideal and non-ideal setting images will help improve the generalization of the model. The model will now be more accurate on the SDD dataset.

Q. At training time, we have access to the entire DBI dataset and a small portion (e.g. 10%) of the SDD dataset but without the SDD labels for this subset.

- This case is tough and similar to Case 1 since we don't have labels. We will have to basically transform 40-50% of the whole dataset randomly. We will add noise, filters, geometric transformation, cropping, resizing and background blur. This will help us to improve accuracy overall including the SDD dataset.

Task 7 –

- The issues would have a significant impact on real-world settings with different use cases.
- There would be a change in performance and accuracy across different settings. This will depend on the similarity between the two settings. The closer they are, the more similar their accuracy is. We noticed throughout our assignment that all models had lower accuracy on the SDD dataset since the models were being trained on DBI, one with an ideal setting like clear background and focused subject whereas, the other was non-ideal.
- The pre-trained models also had an issue of overfitting whereas the CNN models were underfitting. This would be the case as well when we are training and testing on different scenarios.
- To conclude, we learn the significance of using appropriate datasets according to the use case rather than trying to generalize the model on multiple different settings and scenarios.