# SMAI Assignment-4 Q1 Report

Name: Aditya Peketi
Roll: 2024122001

## Question - 1

**wandb Dashboard link**: https://api.wandb.ai/links/adityapeketii-iiit-hyderabad/5kg3ow7i
workspace link - https://wandb.ai/adityapeketii-iiit-hyderabad/SMAI-A4-multitask-fashion-mnist?nw=nwuseradityapeketii

I have implemented a multi-task CNN with a shared convolutional backbone (2/3 Conv-BatchNorm-ReLU-Pool blocks) that branches into separate heads for classification (10 classes) and regression (ink prediction). The model optimizes a joint loss: L=λ1LCE+λ2LMSE.

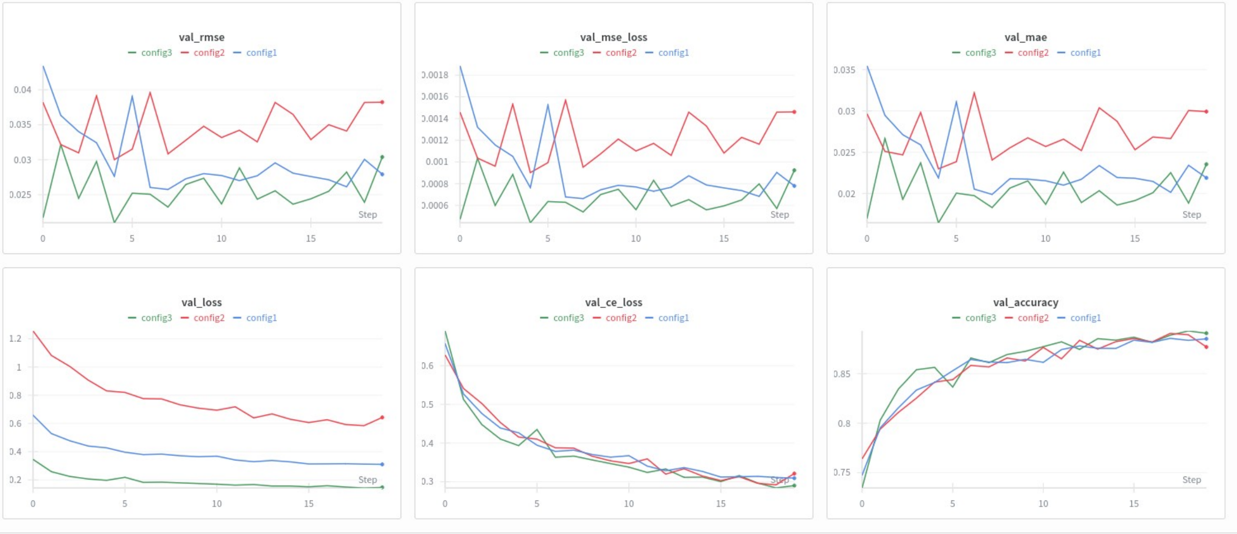
I have tested and logged the model with **10 different configs**:

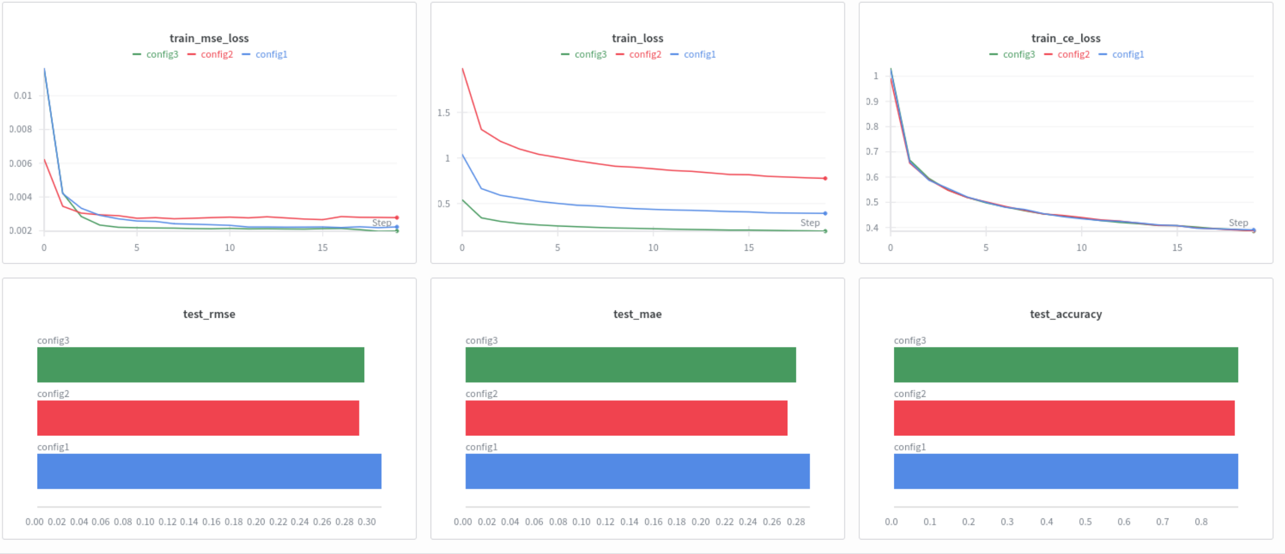
```
configs = [
    {'lambda1': 1.0, 'lambda2': 1.0, 'learning_rate': 0.001, 'dropout_rate': 0.3, 'optimizer': 'Adam', 'batch_size': 128, 'epochs': 20, 'num_layers': 3, 'base_filters': 32},
    {'lambda1': 2.0, 'lambda2': 0.5, 'learning_rate': 0.001, 'dropout_rate': 0.3, 'optimizer': 'Adam', 'batch_size': 128, 'epochs': 20, 'num_layers': 3, 'base_filters': 32},
    {'lambda1': 0.5, 'lambda2': 2.0, 'learning_rate': 0.001, 'dropout_rate': 0.3, 'optimizer': 'Adam', 'batch_size': 128, 'epochs': 20, 'num_layers': 3, 'base_filters': 32},

    {'lambda1': 1.0, 'lambda2': 1.0, 'learning_rate': 0.0008, 'dropout_rate': 0.3, 'optimizer': 'Adam', 'batch_size': 64, 'epochs': 20, 'num_layers': 2, 'base_filters': 16},
    {'lambda1': 1.0, 'lambda2': 1.0, 'learning_rate': 0.0008, 'dropout_rate': 0.3, 'optimizer': 'Adam', 'batch_size': 64, 'epochs': 20, 'num_layers': 2, 'base_filters': 32},

    {'lambda1': 1.0, 'lambda2': 1.0, 'learning_rate': 0.0005, 'dropout_rate': 0.3, 'optimizer': 'Adam', 'batch_size': 64, 'epochs': 20, 'num_layers': 3, 'base_filters': 16},
    {'lambda1': 2.0, 'lambda2': 1.0, 'learning_rate': 0.001, 'dropout_rate': 0.3, 'optimizer': 'Adam', 'batch_size': 128, 'epochs': 20, 'num_layers': 2, 'base_filters': 32},
    {'lambda1': 1.0, 'lambda2': 2.0, 'learning_rate': 0.0008, 'dropout_rate': 0.3, 'optimizer': 'AdamW', 'batch_size': 64, 'epochs': 20, 'num_layers': 2, 'base_filters': 32},
    {'lambda1': 0.5, 'lambda2': 1.0, 'learning_rate': 0.0005, 'dropout_rate': 0.25, 'optimizer': 'AdamW', 'batch_size': 64, 'epochs': 20, 'num_layers': 3, 'base_filters': 16},
    {'lambda1': 1.0, 'lambda2': 0.5, 'learning_rate': 0.002, 'dropout_rate': 0.35, 'optimizer': 'Adam', 'batch_size': 64, 'epochs': 20, 'num_layers': 3, 'base_filters': 16}
]
```

I wanted to test the trade off between classification and regression, thus in the first 3 runs I have only changed lambda1 and lambda2 keeping other parameters constant.





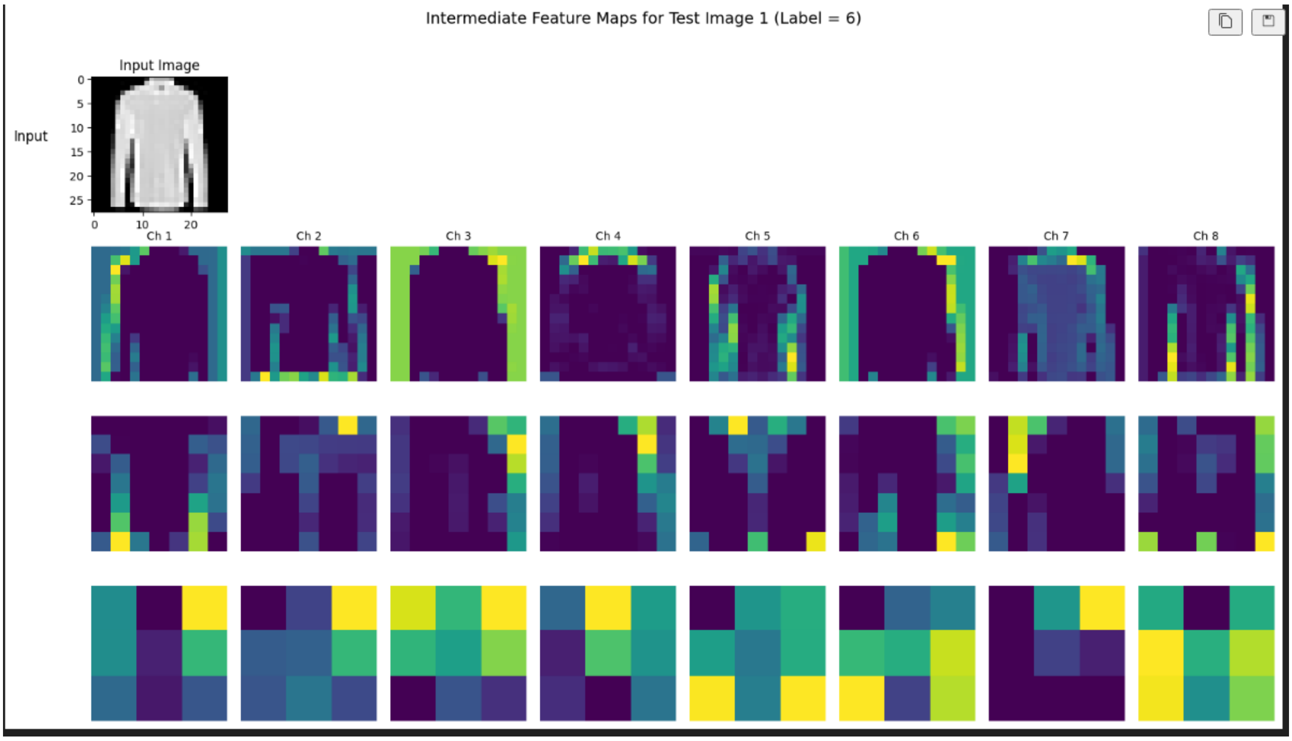
Observations:

When lambda1 is higher than lambda2 model prioritize classification over regression and this should result in a lower CE loss and higher classification accuracies.

And when lambda2 is higher than lambda2 model prioritize regression over classification and this results in a lower RMSE/ MSE loss and a lower classification accuracy

This can be seen from the above plots. The green represent Lambda2 > Lambda1 and red represents Lambda1 > Lambda2 and the blue for both equal.

## Feature Map Viz:



We can see that the first Conv 1 layer local edges of and borders of all the items. We can see bright regions around the contours of the clothing.
In conv layer 2 we can see more fine grain details like sleeves, bag edges and all.

**All the metrics are in the wandb dashboard.**

Best validation accuracy that I got is for config 3 and then config 1 respectively which are **89.083%** and **88.5%**. Other hyper parameters can be checked from the wandb link provided above.

## Question- 2

Assumptions:

- Input Size: H x W (Default: 32x32, Doubled: 64x64)
- Convolutional Layers (Conv2d):
  - Kernel size: k x k
  - Stride: 1
  - Padding: k // 2
- Max Pooling Layers (MaxPool2d):
  - Kernel size: 2x2
  - Stride: 2 (This halves the spatial dimensions)
- Transposed Convolutional Layers (ConvTranspose2d):
  - Kernel size: 2x2
  - Stride: 2 (This doubles the spatial dimensions)
- Parameters: Biases and BatchNorm parameters are ignored in the calculations.
- Symbols:
  - NIC: Number of input channels
  - NF: Number of filters in the first convolutional layer
  - NC: Number of output color classes

---

1. Number of Weights

The number of weights in a convolutional layer is determined by:
out_channels * in_channels * kernel_height * kernel_width

Conv1 (Encoder): NF * NIC * k * k
Conv2 (Encoder): (2*NF) * NF * k * k*

Conv3 (Encoder): (4NF) * (2NF) * k * k
Deconv1 (Decoder): (4NF) * (2NF) * 2 * 2
Deconv2 (Decoder): (2NF) * NF * 2 * 2
Deconv3 (Decoder): NF * NC * 2 * 2
Classifier (1x1 Conv): NC * NC * 1 * 1

Total Weights (Symbolic Formula):
Total_Weights = (NF * NIC * $k^2$) + (2 * $NF^2$ * $k^2$) + (8 * $NF^2$ * $k^2$) + (8 * $NF^2$ * 4) + (2 * $NF^2$ * 4) + (NF * NC * 4) + $NC^2$

Simplified:
Total_Weights = $k^2$ * (NFNIC + 10$NF^2$) + 40$NF^2$ + 4NF*NC + $NC^2$

This formula is valid for both 32x32 and 64x64 input sizes since weights do not depend on input resolution.

---

2. Number of Outputs (Activations)

Number of outputs = total count of activation elements across all layers for a single input image (channels * height * width).

Encoder1: [NF, H/2, W/2] → NF * (H/2) * (W/2)
Encoder2: [2NF, H/4, W/4] → 2NF * (H/4) * (W/4)
Encoder3: [4NF, H/8, W/8] → 4NF * (H/8) * (W/8)
Decoder1: [2NF, H/4, W/4] → 2NF * (H/4) * (W/4)
Decoder2: [NF, H/2, W/2] → NF * (H/2) * (W/2)
Decoder3: [NC, H, W] → NC * H * W
Classifier: [NC, H, W] → NC * H * W

Total Outputs (Symbolic Formula):
Total_Outputs = (NFHW/4) + (2NFHW/16) + (4NFHW/64) + (2NFHW/16) + (NFHW/4) + (NCHW) + (NCH*W)

Simplified:
Total_Outputs = HW * [ NF/4 + NF/8 + NF/16 + NF/8 + NF/4 + 2NC ]
Total_Outputs = HW * [ (4+2+1+2+4)NF/16 + 2NC ]
Total_Outputs = HW * [ 13NF/16 + 2NC ]

Calculations for Specific Input Sizes:
For 32x32 Input (H=32, W=32):
Total_Outputs = 1024 * [ 13NF/16 + 2NC ] = 832NF + 2048NC

For 64x64 Input (H=64, W=64):
Total_Outputs = 4096 * [ 13NF/16 + 2NC ] = 3328NF + 8192NC
(This is exactly 4 times the result for the 32x32 input.)

---

3. Number of Connections (MACs)

The number of connections (or Multiply-Accumulate operations) = (Total Weights in Layer) * (Output Spatial Area of that Layer).

For encoder Conv2d layers, computation happens before MaxPool2d.

Conv1: Weights = $NFNICk^2$, Output area = $HW$ → Connections = $(NFNICk^2)(HW)$
Conv2: Weights = $2NF^2k^2$, Output area = $(H/2)(W/2)$ → Connections = $(2NF^2k^2)(HW/4)$
Conv3: Weights = $8NF^2k^2$, Output area = $(H/4)(W/4)$ → Connections = $(8NF^2k^2)(HW/16)$
Deconv1: Weights = $8NF^24$, Output area = $(H/4)(W/4)$ → Connections = $(32NF^2)(HW/16)$
Deconv2: Weights = $2NF^24$, Output area = $(H/2)(W/2)$ → Connections = $(8NF^2)(HW/4)$
Deconv3: Weights = $4NFNC$, Output area = $HW$ → Connections = $(4NFNC)(HW)$
Classifier: Weights = $NC^2$, Output area = $HW$ → Connections = $NC^2(H*W)$

Total Connections (Symbolic Formula):
$\text{Total\_Connections} = HW * [\ NFNICk^2 + 2NF^2k^2/4 + 8NF^2k^2/16 + 32NF^2/16 + 8NF^2/4 + 4NFNC + NC^2\ ]$

Simplified:
$\text{Total\_Connections} = HW * [\ NFNICk^2 + 0.5NF^2k^2 + 0.5NF^2k^2 + 2NF^2 + 2NF^2 + 4NFNC + NC^2\ ]$
$\text{Total\_Connections} = HW * [\ k^2NF(NIC + NF) + 4NF^2 + 4NF*NC + NC^2\ ]$

Calculations for Specific Input Sizes:
For 32x32 Input (H=32, W=32):
$\text{Total\_Connections} = 1024 * [\ k^2NF(NIC + NF) + 4NF^2 + 4NF*NC + NC^2\ ]$

For 64x64 Input (H=64, W=64):
$\text{Total\_Connections} = 4096 * [\ k^2NF(NIC + NF) + 4NF^2 + 4NF*NC + NC^2\ ]$
(This is exactly 4 times the result for the 32x32 input.)

## Hyperparameter Tuning – Best Run

*Validation Accuracy = 50.10645%*
*Validation Loss = 1.3*

*hyperarams - {'learning_rate': 1e-3, 'batch_size': 32, 'NF': 32, 'kernel_size': 5, 'optimizer': 'Adam'}*

***Link*** **-** https://wandb.ai/adityapeketii-iiit-hyderabad/SMAI-A4-image-colorization
**Best model -** https://wandb.ai/adityapeketii-iiit-hyderabad/SMAI-A4-image-colorization/runs/9lcfmg4d