Precog Task Documentation and Report:

Task name: The Lazy Artist (Computer vision)
Author name: Balasubramanian K
Roll Number :2025121002
Branch: LCD

Tasks i have done:
Task 1,2,3,4,5,6(partial)

Note to the reader/evaluator:

I have stored the results across several colab python notebook files and they are reproducible.
Often times in the notebook i may have referred to certain notebook links.These links contain the visualizations and the outputs i talk about in the report
In an effort to keep the report concise and to the point i have avoided pasting results images as they are easily available in the linked notebooks to cross check

Why this document is important?
The document here contains a jist of all my experiments ,their results and the inferences i drew from them be it successful with regards to a task or not
Without this document my work is just code without explanations

Note 2: the repository to which i refer inside which model weights are available are present in my github

TASK 0: Objective: coloured train and test MNIST dataset.

Idea 1: To colour the dataset,assign 1 RGB value to each digit and across the dataset wherever the label was same,colour the digit with assigned RGB value.Use the intensity values of original grayscale dataset to colour the pixel values.Link to notebook

Drawback : the non digit parts of original dataset had intensity values 0.hence no background colour

Idea 2: Introduce some background colour with fixed intensity and some gaussian noise to make the background more realistic.

Outcome: The coloured images looked better but i felt it could be made more realistic
Link to notebook

The Improvement: after playing around with intensity of noise and adding foreground gaussian noise ,the images looked good enough to me

Reproducibility: to keep the experiments regeneratable,the probability numbers were picked from a pre-determined set of same set.

Easy Train Set: if the numbers picked from 0-1 less than 0.95, the assigned colour was coloured or else any random colour.

Hard Test Set: Initially randomized colours were applied to digits with no correlation

link to completed task0 notebook

TASK 1: Objective: Train a simple model on this MNIST dataset

Implementation 1: unsure of what exact CNN architecture to use(number of layers,convolutions,pooling and nodes per layer) for the dataset i decided to use the one provided in official pytorch documentation it was meant for single channel(greyscale) MNIST,i modified it to accommodate the 3 RGB channels.The model had 2 convolutional layers:
Layer 1:-> input : 3 channels(RGB),output: 32 feature maps,operators=3x3 kernels
Layer 2:-> input : 32 channels,output : 64 feature maps, operators=3x3 kernels
After each convolution ReLu is applied upon feature maps and Relu is performed upon feature maps and after the last convolution,maxpooling of 2x2 is done
The neural network consisted of 2 layers:
Input layer: 9216(=64x12x12) inputs
Hidden layer: 128 nodes
Output layer: 10 classification nodes(corresponding to each digit)

The issue: The classifier model performs too well on the easy train set(99.9%) as well as the hard test set(93%) . It is an overkill maybe due to large number of feature maps leading to some of them learning shapes as well .The link to the colab notebook

Modification 1: i decided to simplify CNN architecture to
Layer 1:-> input : 3 channels(RGB),output: 24 feature maps,operators=5x5 kernels and stride=2
Layer 2:-> input : 24 channels,output : 48 feature maps, operators=3x3 kernels and stride=1

Performance: The test set(Hard) still had an high accuracy but reduced to 73% .reducing feature maps seems promising.Click here for the notebook

Modification 2: i further simplified the CNN to
Layer 1:-> input : 3 channels(RGB),output: 16 feature maps,operators=7x7 kernels and stride=7
Layer 2:-> input : 16 channels,output : 24 feature maps, operators=3x3 kernels and stride=1
And added an extra pooling layer after conv1.

This seemed to solve the problem. With the test set accuracy having come down to 18%.(train accuracy stayed 96%) link to the colab notebook

However looking at the confusion matrix there seemed to be a significant turquoise 8 bias.was it an 8 or an turquoise bias,click here to see my hypothesis and investigation on why this may have happened.

This led me to modify hard test set (from task 0) to give it an inverted correlation where each digit is assigned to the colour of the next digit(in train set).the confusion matrix of this will show for definite if the model is looking at the colour.

And the confusion matrix did change as I expected showing the colour bias as well the test accuracy dipping even further to 12%(close to random guessing). Link to final Task1 notebook

TASK 2: Objective: Find out what the neuron sees

Implementation 1: As mentioned in the task doc,I took a an optimizable tensor(random tensor) of same shape as mnist dataset images (3x28x28) and upon performing gradient ascent (the fancy word for keeping the weights same in a model and changing input to maximize a neuron activation) on both convolutional layers conv1 and 2 channelwise and taking the mean of each neuron for that channel , the outputs(16 channels for conv1 and 24 for conv2) seemed like random stripped and dotted patterns of colour combinations on conv1 and almost specific set of coloured noise like on layer 2.

Possible issue: The neuron activation was conducted on a small size of the CNN with only 2 conv layers whereas the openAi microscope and similar models were analysed on models with several layers (Deep neural nets)

However the one inference i could make was that the conv1 layer seemed to have 3 channels almost completely green and 2 blue,prompting to a possible explanation of the turquoise bias in the randomized test set discussed earlier

I couldn't find any patterns as described by this article on feature visualization . but inspired its content ,i decided to look at data points which maximised the neuron channel activations the most.

And as expected of the bias model the top activations in channels of Conv1 seemed to close to universally be only affected by the colours

In layer 2 some neuron channel seemed to exhibit polisemanticity particularly channel 0 which had among its top activations different digits of different colours and channel 1 and 8 seemed to be activated by 1s of different colour suggesting it may be activated by a stroke in the middle

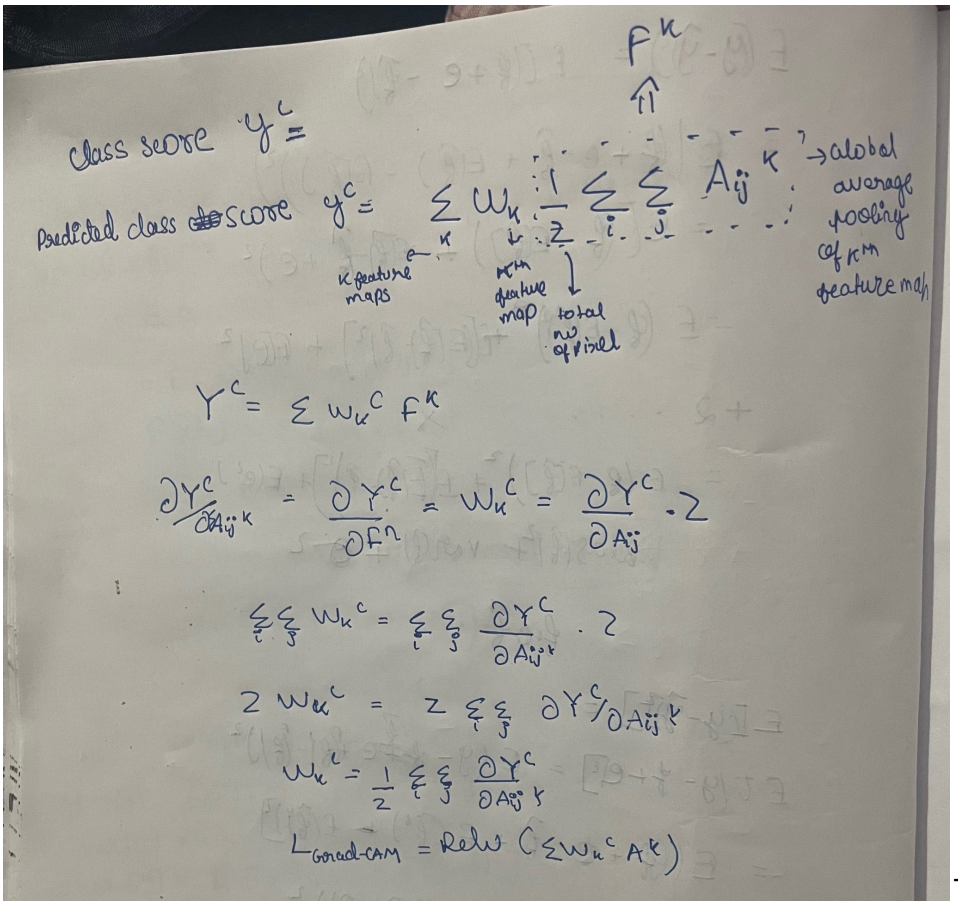
Apart from channel mean activation,Raw activation of a neuron from a channel was also tested(by raw neuron we mean to a particular pixel from the 5x5(after 2 convs) pixel per channel).This seemed to provide better results.The channel 5's (2,2) pixel seemed to show why channel 5 in general preferred a turquoise 5,with the out lay being clear in the raw activation

The raw spatial visual activations which included taken mean activation of all neurons at a particular location in a channel however did not seem give any tangible inferences.

Click here for all the visualizations in the notebook

TASK 3: Objective: Training a Grad-Cam to see where the model focuses,

The theory:
Grad-CAM generates a heatmap showing image regions that most influence a model's decision.



The process followed was:
1. Perform a forward pass to obtain predictions
2. Compute gradients of the predicted class with respect to the final convolutional feature maps
3. Average these gradients to obtain channel importance weights
4. Combine these weights with the feature maps
5. Apply ReLU and upsample to image size t

The practical:
In order to save the the gradients of final conv layer , we use pytorch hooks.Link to the notebook implementation

The Result: On a biased train set image the Gradcam heat map smears across the entire image showing it doesn't focus on strokes and lines of the digit
On a test set image, it seems to be looking at regions irrelevant to the digit where more of the colour background seemed to be present.

References : an amazing video an equally good article

TASK 4: Objective: Train a model with at least 70% accuracy on the hard set(colour debiasing)

A Thought:
“how will a baby classify the hard set having only seen the easy set in its life?”

A tentative solution:
”the parents should have told the baby to differentiate by seeing the shape (which in turn is visible because of the contrast between background and foreground pixels)”

And an attempt:
hypothesis:->essentially i postulated that if i can detect the contrast through edge detection of sobel filters and some how reward the model for detecting based on edges the model will become robust.
The loss function: if F is the set of feature maps in the last convolutional layer,
$A=(1/c)x(\sum Upsample(F))$ where c is number of channels in conv2
E=sobel(X) X is input image
$L_{edge} = \ // A-E //^2$
$L_{total}=L_{cross\ entropy}+ lambda(L_{edge})$
However by the 4th epoch it could only perform with 5% accuracy on the hard test set(worse than random).No amount of hyper parameter tuning had worked.I suppose the colour bias was too much to for the model to ignore.(after all neural networks are just maths and babies deserve more credit than we give them). Notebook_link to the failed model(check only training epoch logs)
As a last ditch effort i also tried to see if differences between gradcam highlights and sobel edge detection can be minimized,it did not work again.Notebook for this

Possible reason why it did not work: the colour bias was too strong.a successful model should punish the colour bias-> Hypothesis 2

Another attempt
So inorder to punish the strong colour bias i decided to define a loss
$Lfeat\_color=mean(|F(R)-F(G)|+|F(G)-F(B)|+|F(B)-F(R)|)$ where F(i) are activations originating from each channel. It however failed with initial success (31% after epoch 2)

After hyperparameter tuning i abandoned this loss function too and i tried a new one:

$G=dL_{cross\ entropy}/dX(c)$ where c represents the colour channels
We get $G_R, G_B\ G_g$ and we define $L_{colour}=mean(G_R-G_B)+mean(G_g-G_B)+mean(G_R-G_g)$

$L_{total}=L_{cross\ entropy}+ lambda(L_{colour})$

This loss gave a colour unbiased model with a test accuracy of 91%.From henceforth i will call this model the colourloss model.

The link to this colourloss model notebook

Method 2:For this i simply decided to implement this paper “Improving CNN Robustness to Color Shifts via Color Balancing and Spatial dropout”

There are two parts to it:
The Color Balancing layer mixes information across the RGB channels using a weighted linear combination. This reduces the dominance of any single color channel and weakens the direct link between a specific color and a digit class. As a result, the network is encouraged to focus more on intensity patterns and structural details of the digit rather than relying on color alone.

The Spatial Dropout layer further improves this by randomly removing entire feature maps during training. This prevents the model from over-depending on a single feature channel that might encode color-based shortcuts. Instead, the model is pushed to learn more generalizable shape-based features.

This model however only got 72% accuracy(just pulled past the requirements) in hard test set (99% in easy train set)

The link to the Augmented consistency model notebook

Another mode(Not really)l:

The initial pytorch documentation based model gets 74% hard test set accuracy without modifying any training step.This is mostly because its a more robust CNN and with 9126 input channels therefore its like throwing an hammer at an ant and also not really makes the changes in the training which is cheating

The link to this cheater hammer model notebook

TASK 5: Objective: perform a targeted adversarial attack on the models by deceiving it to predict a 7 as a 3.

Hypothesis : finding the average pixel value of 3 and shifting the image of a 7 towards the average 3 with per pixel change from 0 to 0.05 (max) will yield the inputs 7's predicted class to slowly become 3

Mini hypothesis: the average values of 150 train set 3s will be practically same as average value 1000 train set 3 and therefore the entire train set 3s (6000)

The mini hypothesis is proved inside the notebook for task 5.However the main hypothesis doesnt hold true.The minimum e change required for it to be true is 0.28 (71/255 intensity points) at which the overlay of 3 is clearly visible

Analysis : The CNN has a non linear function predicting values,it may be possible without trying to make the entire 7 change towards an average 3.

Study: watching a video on targeted adversarial attack and the toronto university tutorial on one led me to implement a projected gradient descent`(PGD) attack on the models where the required class prediction score is taken as loss.

Results
Interestingly the Biased model and the and augmented consistency based models require way more push than the colorloss model which relatively scumbs to small changes in epsilon(0.021) over fewer steps(4 steps) for the examples.I suspect this may be due to the other models being heavily colour biased,their neurons leading to a 3 may not be activated without the input image being the biased colour.The colour biased model takes around 100000 steps with 0.05 epsilon change(max allowed change)

The link to the notebook

TASK 6: Objective: Train an sparse auto encoder to decompose intermediate hidden layers and label any meaningful features present and changing these sparse auto encoded features(in the biased model) to see the changes in predictions

Disclaimer: I have not done part 2 of the task and only implemented the SAE on the biased model and not on the rectified colourloss model and Augmented Consistency model

Method:

1. We collect activations vectors from the 2nd convolution layer(Conv2) and each channel of each image is transformed into a high dimensional feature vector of shape [number of images, ChannelsxHeightxWidth of conv2

2. Next i trained a sparse Autoencoder (partially taken from here) with a a loss function Loss=reconstruction_loss + lambda * sparsity penalty.The sparsity penalty forces the features to be selective and the reconstruction loss focuses on information retention

3. I visualized SAE feature weights by reshaping them into 5x5 Conv2 feature map shapes and taking the mean across channels gives a spatial sensitivity map

4. Found the corresponding Top activating images for select few SAE feature and labelled (based on visual perception of the spatial sensitivity maps and the activation images) them appropriately.(You can find those features in the end of my notebook for the task)

Major findings:

Few features(3 to be exact of the 12 i inspected) could be clearly correlated to some “human understandable” labels however i was unable to label several other SAE feature maps and their top activations i inspected.

However this at least shows that partially the super positions of distinct features or polysemanticity of neurons can be decomposed when given large feature space by the SAE model as highlighted by the paper mentioned

Link to the video i saw for intuition