Satheesh D M
MA24M023
ma24m023@smail.iitm.ac.in
+91-9150288376
24/08/2025

CountAI - Internship Interview Assessment Task Report

Problem Statement Given:

To build an AI model capable of accurately distinguishing defective from non-defective images. The candidate may employ any approach they find most suitable and effective. (Dataset)

Aim : To classify the given images into defects and non-defect classes.

- I am attempting to classify the images into 5 given classes.
  Approach used:
  - Multi class (5) classification
  - Transfer Learning
  - Fine tuning at last

About the dataset : It has 5 classes with 62 images of a piece of textile in total. Highly imbalanced dataset, otherwise addressed in the literature as a dataset with **"long tailed class imbalance".**

- non defect (50 samples),
- defect {hole (2), lycra cut (2), needln (5), twoply (3)} (12 samples)

Problems in the dataset to be addressed:

Issue in the dataset	Mitigated by
• Very less data points to train any AI model (ML/DL based). • Almost very similar datapoints. (Same textile, colour, viewing angle, etc)	• Data Augmentation • Transfer learning + Finetuning
• Highly class imbalanced dataset.	• Re-weighting loss function. • Random Weighted sampling of data.
• Image size (1270 x 720 px) very close to standard **16:9 aspect ratio.**	• Learn a custom head in Deep learning approach.

Data Augmentation Strategy: (`ANYTING_IN_GREEN` = Refer to code "augmentations.py")

Step 1: Data Collection

- Load raw images from class-specific subfolders (`RAW_DATA_DIR`).
- Supported formats: `.jpg`, `.jpeg`, `.png`.

Step 2: Dataset Splitting (before augmenting to avoid **data leakage**)

- Reserve a small set of original images for validation (`VAL_ORIGINALS`).
- Both the train and validation splits go into their respective class in the finally splitted dataset.

  # (A **unique test set** may be generated by changing the seed value of augmentation pipeline)

Step 3: Augmentation Operations *(probability-driven)* (Separate for train split and validation split)

- Geometric:
  - Horizontal/vertical flips
  - Small rotations (±5° only → avoids over-rotation)
  - 180° rotation (upside-down check)
  - Zoom in/out (no stretching/squeezing)
  - Translations (small shifts)
- Photometric:
  - Brightness, contrast, sharpness changes
  - Hue & saturation shifts
  - Gaussian blurring
- Noise:
  - Additive Gaussian noise

Step 4: Probabilistic Control

- Each augmentation applied conditionally with class-specific probabilities (`PROBS`).
- Prevents unrealistic distortions and keeps variability balanced.

  # (There may be similarities but only very infinitesimal probability of having duplicates)

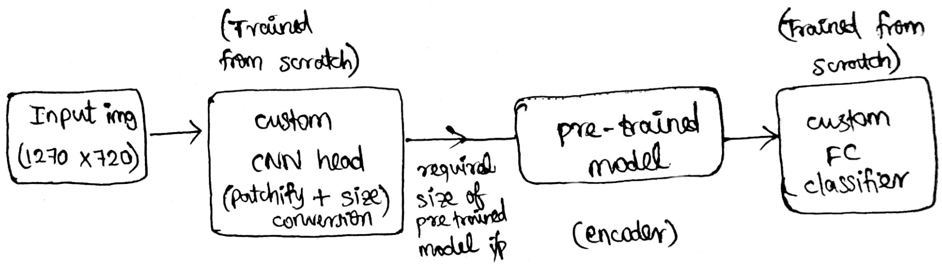
Step 5: Data Balancing

- Generate a fixed number of augmented samples per original image (`PER_ORIGINAL_COUNTS`), according to calculations made to maintain a 80:20 ratio b/w train and validation sets.
- Equally augmented the dataset preserving the original distribution.

Comments : (In reference to final submissions not baseline modelling)
- With this pipeline, the dataset is scaled from 62 images into a (62 * 200 =) **12.4k unique images.**
- Large enough size to finetune a decent pre-trained deep neural network model.

- But appropriately scaled to maintain original dataset distribution. So still there is a **class imbalance problem** to be addressed.

Multiclass classification Model (Components):



Baseline modelling: **(Aim : To verify if this approach is working.)**

- **Scratch training didn't work:** Tried it, the metrics were very unstable.

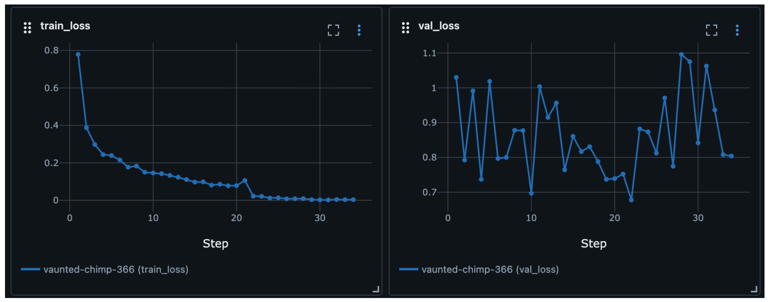
With the following configurations an attempt was made to setup a baseline model based on finetuning,

- **Augmented Data :** 14.7K total samples (original defects * 600 + original no_defects * 150)
- **Random Sampler :** Inverse frequency weighting random sampler for data loader.
- **Loss function :** Class Balanced Cross entropy loss (inverse frequency weighting).
- **Optimizer :** Adam
- **Method :** Scratch training Head & Classifier only by freezing pre-trained encoder weights for the first 20 epochs. Then unfreeze only the last 2 layers of encoder and finetune it with a small learning rate (1e-4).
- **Performance based LR scheduler :** ReduceLROnPlateau
- **Metrics :** f1 macro, f1 weighted, accuracy.
- **Model Tracking :** Integrated MLflow with the training pipeline.
- **Custom Head:** 2 CNN layers with strides 2 & kernel sizes 7 and 5 respectively + bilinear upsampling to match input size of the pre-trained encoder used with 3 channels (@ i/p & o/p)
- **Custom Classifier:** 2 FC layers with 256, 128 neurons (ReLu, Batchnorm) connected with 5 neuron softmax last layer.
- **Regularization:** Early stopping, Batchnorm & Dropouts b/w layers in custom head and classifier.

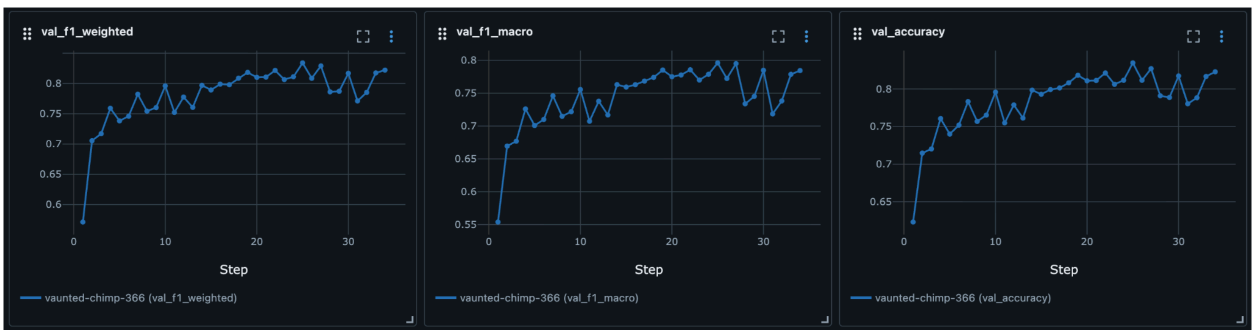
Encoder Models (small version)	Total whole parameters	Best val f1 weighted	Best val f1 macro	Best val accuracy
ResNet18	11.25 M	0.8338	0.7950	0.8345
ShuffleNet_V2	1.4 M	0.8035	0.7742	0.8062
SqueezeNet1	0.8 M	0.7330	0.6735	0.7452
EfficientNet_B0	4.1 M	0.7857	0.7478	0.7904
MobileNetV3	1 M	0.7259	0.6655	0.7390

<u>Observations:</u> (Explained with Resnet18 Example. Similar for all)

- Smooth training loss curve but Spiky validation loss curve.
- Scale mismatch between training and validation loss
  Possible reasons: Unstable updates, often caused by a few very "hard" examples from minority class. Pushing the model of the rails, train & validation set data distribution mismatch.



- Plateauing of validation metrics, not going down and indicating overfitting. Instability in metrics.
  Possible reasons: lack of robust feature learning.



- F1 weighted and F1 macro are having a gap between their values, this means unaddressed class imbalance.
- Training loss and metric curves are smooth indicating that training happens.
- Time Taken on average : **4-5 hrs for every model fine tuning.**

<u>Problems to address :</u>

- Train & Validation Data distribution mismatch.
- Class imbalance within any given split (train/validation)
- Perspective change and over blurring during augmentation causing important features to fade.
- Not learning Robust features.
- Could not differentiate between overfitting and proper robust feature learning.

<u>Post Baseline :</u> **Corrections Adopted from the baseline modelling:**

- **Changed augmentation strategy** and scaled the whole dataset 200 times as whole. But each individual sample is scaled in accordance with the calculation made to maintain a stratified **80:20** train validation split.
- **Removed perspective transformation** causing anomalies to deform.
- **Reduced** blurring and noise parameters in augmentation.
- Changed to **Class Balanced Focal Loss** (Effective number weights)
- **Effective number** based random sampler for addressing class imbalance.
- Changed Classifier into a **scale invariant cosine classifier** (refer code) for learning robust features.

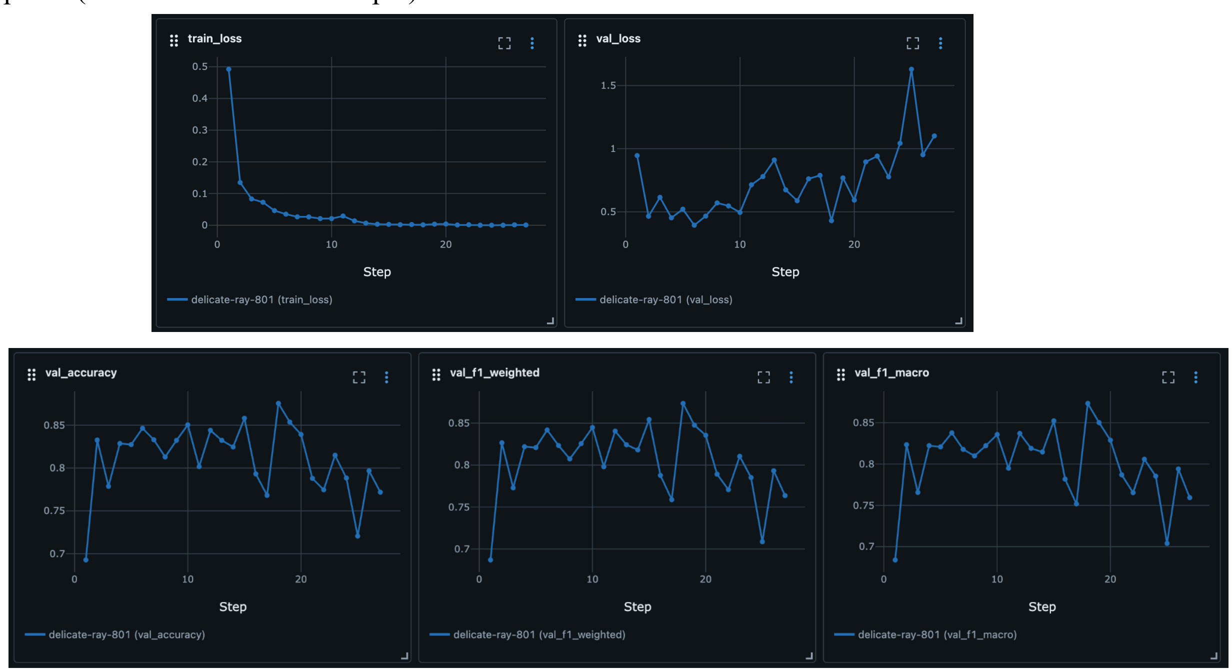
- Fixed unfreezing of encoder last layers epoch as 10.
- Increased Early stopping counter to max at 9 epochs. (Accounting for LR scheduler to kick in this time)

Reference: (Highlighted the points referred to) [Deep Long-Tailed Learning: A Survey]

Results Table after Adjustments: (Without any hyperparameter tuning)

Pre-Trained Encoder Models (small version)	Best f1 weighted (baseline)	Best f1 weighted (post_baseline)	Best f1 macro (baseline)	Best f1 macro (post_baseline)
ResNet18	0.8338	0.8754	0.7950	0.8733
ShuffleNet_V2	0.8035	0.8129	0.7742	0.8098
SqueezeNet1	0.7330	0.7067	0.6735	0.7020
EfficientNet_B0	0.7857	0.7232	0.7478	0.7202
MobileNetV3	0.7259	0.6817	0.6655	0.6738

Sample plots: (ResNet18 encoder example)



Comments on Post baseline corrections made :

- Clearly from the validation loss curve, the model overfits and the oscillations remains but less compared to before (maybe not enough unique variety of augmentations to learn from)
- But all the validation metrics (f1_macro, f1_weighted, val_accuracy) are close together. This means the class imbalance is addressed properly.
- But metric values improved mostly as whole (scale invariant cosine classifier seems to work)

Final Submission model : (refer in MLflow UI) (Also refer **README.md** file)

- Experiment name : **post_baseline_ResNet**
- Experiment id : **727670261461040353**
- Run Name : **delicate-ray-801**

Future ideas:

To try advanced data augmentation techniques like SMOTE and GAN. However, the input image dimension of 1270 × 720 makes it difficult to integrate them into the existing pipeline within the given time frame.

Approach for future:

- Augment the original 62 images with traditional augmentations to build a dataset of about 12.4K images.
- Then use SMOTE or GAN to further grow the training set in a more unique way.

Problems:

- SMOTE:
  - Most embeddings work with inputs around 256 × 256.
  - Cropping a patch of this size from the non_defect class is easy, but centering and cutting such a patch from the already augmented (warped) training dataset is problematic.
  - One workaround is to cut a 256 × 256 patch, apply SMOTE, and then patch it back into the larger image - but this is tricky.
- GAN:
  - Faces the same cut–create–patch issue as SMOTE.
  - Additionally, fine-tuning a GAN on the training set is time-consuming.

Other directions to reduce overfitting:

- Try more aggressive regularization-based augmentations:
  - Random Erasing
  - Random local rotation
  - Different noise injection
- Add L2 regularization to smooth the validation curves.

References used for Future ideas section : (Highlighted the points referred to)

- [Data Augmentation in Classification and Segmentation: A Survey and New Strategies]
- [A survey on Image Data Augmentation for Deep Learning]