

# 3LC x AWS Hackathon - Final Write-Up

## Chihuahua vs. Muffin - A Data-Centric Binary Image Classification Pipeline Using 3LC

## 1. Introduction

This project implements a binary image classification system for distinguishing between Chihuahuas and Muffins using a data-centric AI workflow powered by 3LC. Instead of focusing on model complexity, the goal was to steadily improve classification performance by improving data quality, label accuracy, and dataset structure through iterative refinement.

The workflow leverages 3LC's capabilities - including embeddings, confidence-based filtering, misclassification analysis, clustering, and table versioning - to build a clean, consistent, and highly informative training dataset.

## 2. Baseline Setup

We began with:

- 200 labeled images (Chihuahua / Muffin)
  - 4,533 unlabeled images under the undefined class
  - ResNet-18 (no pretrained weights)
  - Basic augmentations
  - 20 training epochs, learning rate 0.0001

This created the baseline accuracy, serving as the foundation for all subsequent data-centric improvements.

**Initial Accuracy: 84.80%**

```
Terminal Local Local (2) +  
spurts of [OK] 2020-09-04 10:00:00...  
Validation Epoch 0/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.6732] 74/74 [00:06:00:00, 11.23it/s, val_loss=0.6732]  
[OK] Best model saved to 'resnet18_classifier_best.pth'  
Epoch 1/10 - Val Loss: 0.4393, Val Accuracy: 83.11% | 13/13 [00:00:00.00, 4.64it/s, loss=0.6732] 74/74 [00:06:00:00, 11.23it/s, val_loss=0.6732]  
[OK] New best model! Validation accuracy: 83.11% | 13/13 [00:00:00.00, 4.64it/s, loss=0.6732] 74/74 [00:06:00:00, 11.23it/s, val_loss=0.6732]  
Epoch 2/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.5538] 74/74 [00:06:00:00, 11.24it/s, val_loss=0.267]  
Validation Epoch 2/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.5538] 74/74 [00:06:00:00, 11.24it/s, val_loss=0.267]  
Epoch 3/10 - Val Loss: 0.5338, Val Accuracy: 82.26% | 13/13 [00:00:00.00, 4.64it/s, loss=0.5538] 74/74 [00:06:00:00, 11.24it/s, val_loss=0.267]  
Epoch 4/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.5338] 74/74 [00:06:00:00, 11.25it/s, val_loss=0.368]  
Validation Epoch 4/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.5338] 74/74 [00:06:00:00, 11.25it/s, val_loss=0.368]  
Epoch 5/10 - Val Loss: 0.4409, Val Accuracy: 83.53% | 13/13 [00:00:00.00, 4.64it/s, loss=0.5338] 74/74 [00:06:00:00, 11.25it/s, val_loss=0.368]  
[OK] New best model! Validation accuracy: 83.53% | 13/13 [00:00:00.00, 4.64it/s, loss=0.5338] 74/74 [00:06:00:00, 11.25it/s, val_loss=0.368]  
Epoch 6/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.5338] 74/74 [00:06:00:00, 11.25it/s, val_loss=0.368]  
Validation Epoch 6/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.5338] 74/74 [00:06:00:00, 11.25it/s, val_loss=0.368]  
Epoch 7/10 - Val Loss: 0.3960, Val Accuracy: 84.80% | 13/13 [00:00:00.00, 4.64it/s, loss=0.5338] 74/74 [00:06:00:00, 11.27it/s, val_loss=0.0765]  
[OK] New best model! Validation accuracy: 84.80% | 13/13 [00:00:00.00, 4.64it/s, loss=0.5338] 74/74 [00:06:00:00, 11.27it/s, val_loss=0.0765]  
Epoch 8/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.3960] 74/74 [00:06:00:00, 11.27it/s, val_loss=0.0765]  
Validation Epoch 8/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.3960] 74/74 [00:06:00:00, 11.27it/s, val_loss=0.0765]  
Epoch 9/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.3960] 74/74 [00:06:00:00, 11.27it/s, val_loss=0.0765]  
Validation Epoch 9/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.3960] 74/74 [00:06:00:00, 11.27it/s, val_loss=0.0765]  
Epoch 10/10: 100% | 13/13 [00:00:00.00, 4.64it/s, loss=0.3960] 74/74 [00:06:00:00, 11.27it/s, val_loss=0.0765]  
Epoch 10/10 - Val Loss: 0.3294, Val Accuracy: 83.45% | 13/13 [00:00:00.00, 4.64it/s, loss=0.3960] 74/74 [00:06:00:00, 11.27it/s, val_loss=0.0765]  
Epoch 10/10 - Val Loss: 0.3294, Val Accuracy: 83.45% | 13/13 [00:00:00.00, 4.64it/s, loss=0.3960] 74/74 [00:06:00:00, 11.27it/s, val_loss=0.0765]  
*****  
Training complete!  
Best validation accuracy: 84.80%  
*****  
  
[OK] Loaded best model for metrics collection  
[OK] Best model saved to 'resnet18_classifier_best.pth'  
  
Collecting metrics on train set with best model...  
Collecting 3LC metrics for dataset chinatown-muffin  
100% 0:00:30  
  
[OK] Metrics collection complete!  
  
Reducing embeddings using PaCMAP...  
WARNING:pacmap.pacmap:Note: n_components >= 2 have not been thoroughly tested.  
Embeddings Reduction complete!  
  
[OK] All done! View results at https://dashboards.3lc.ai  
PS C:\Users\juheel\Desktop\hackathon>  
Hackathon 1: startet_notebook.ipynb  
1 1 CRLF UTF-8 4 spaces Python 3.12
```

### 3. Data-Centric Iterative Loop

Our process followed the structured data-centric loop:

Train → Analyze → Fix → Retrain

Across 15 dataset versions, we consistently used model feedback, confidence metrics, embedding clusters, and misclassification reasoning to improve label quality and dataset balance.

## 4. Version-by-Version Dataset Improvements

## Version 1 — High Confidence Auto-Labeling (>0.95)

High-confidence unlabeled samples were automatically assigned Chihuahua/Muffin labels. This expanded the dataset while maintaining precision.

## Version 2 — High-Loss Sample Removal

Samples with high loss (~21.7) were identified and removed or corrected, reducing noisy and harmful examples.

## Versions 3–7 — Iterative Labeling Using Confidence (>0.94 / >0.95)

Across multiple passes:

- Predicted labels were applied to unlabeled samples.
- ID-based filtering was performed.
- Outliers were re-labeled as undefined.

The dataset steadily grew with high-quality additions.

## Versions 8–9 — Strict Threshold Labeling (>0.99)

Only extremely confident predictions (>0.99) were accepted. These samples were manually validated. Ambiguous or visually odd samples remained undefined.

## Versions 10–11 — Low Confidence Mining (<0.68 / <0.70)

- Embedding plot colored by confidence
- Low-confidence clusters were corrected
- Mislabeled and inconsistent images were removed or fixed

## Hyperparameter Improvements

- Epochs increased from 20 → 50
- Learning rate increased to 0.001

## Mid-Pipeline Accuracy (after Iteration 10–11): 91.98%

The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure, including a 'Hackathon' folder containing 'content' (with 'api.png', 'dashboard.png', 'service.png', 'task128.png'), 'chihuahuas', 'muffin', and 'train128'. The right sidebar shows file navigation and a code editor with 'train.py'. The terminal at the bottom shows the training process:

```
Epoch 16/20: 100% | Validation Epoch 16/20: 100%
Epoch 16/20 - Val Loss: 0.3385, Val Accuracy: 90.96%
Epoch 17/20: 100% | Validation Epoch 17/20: 100%
Epoch 17/20 - Val Loss: 0.4483, Val Accuracy: 85.54%
Epoch 18/20: 100% | Validation Epoch 18/20: 100%
Epoch 18/20 - Val Loss: 0.3941, Val Accuracy: 88.20%
Epoch 19/20: 100% | Validation Epoch 19/20: 100%
Epoch 19/20 - Val Loss: 0.3396, Val Accuracy: 90.88%
Epoch 20/20: 100% | Validation Epoch 20/20: 100%
Epoch 20/20 - Val Loss: 0.3517, Val Accuracy: 91.98%
[OK] New best model! Validation accuracy: 91.98%
=====
Training complete!
Best validation accuracy: 91.98%
=====
[OK] Loaded best model for metrics collection
[OK] Best model saved to 'resnet18_classifier_best.pth'
Collecting metrics on train set with best model...
Collecting SLC metrics for dataset chihuahua-muffin
66% 0:08:12
```

## Version 12 — Cluster-Based Label Repairs

Low-confidence clusters were corrected, mislabels were fixed, and similar examples were grouped.

## Version 13 — Reassigning Undefined Samples

Undefined samples were inspected using predicted label, confidence, and cluster location. Confident predictions were promoted to true labels.

## Augmentation Upgrades

To improve generalization:

- RandomRotation
- ColorJitter
- RandomResizedCrop
- RandomHorizontalFlip

## Version 14 — Outlier Removal & Fine Cluster Cleaning

Removed noisy images, incorrect groupings, and visually inconsistent samples.

## Version 15 — Final Dataset Cleanup + 50 Epoch Training

A final sweep ensured all labels were correct. The final model was trained with:

- 50 epochs
- Strong augmentations
- Dynamic learning rate scheduling

## Final Accuracy: 95.61%

The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure with folders like Hackathon, ipynb\_checkpoints, 3c-env, content, and train128. The main window has three tabs: README.md, starter\_notebook.ipynb, and train.py. The train.py tab shows Python code for training a ResNet18 classifier. The terminal tab at the bottom shows the command-line output of the training process, which includes epoch logs, validation metrics, and a summary indicating a final accuracy of 95.61%.

```
EPOCHS = 50
BATCH_SIZE = 16
LEARNING_RATE = 0.005

# Project configuration
PROJECT_NAME = "Chihuahua-Muffin"
DATASET_NAME = "chihuahua-muffin"

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f"Using device: {device}")

# MODEL DEFINITION
class ResNet18Classifier(nn.Module): ...
```

```
Epoch 45/50 - Val Loss: 0.1860, Val Accuracy: 93.24%
Epoch 46/50: 100%
Validation Epoch 46/50: 100%
Epoch 46/50 - Val Loss: 0.2027, Val Accuracy: 91.72%
Epoch 47/50: 100%
Validation Epoch 47/50: 100%
Epoch 47/50 - Val Loss: 0.2024, Val Accuracy: 91.04%
Epoch 48/50: 100%
Validation Epoch 48/50: 100%
Epoch 48/50 - Val loss: 0.1944, Val Accuracy: 92.91%
Epoch 49/50: 100%
Validation Epoch 49/50: 100%
Epoch 49/50 - Val Loss: 0.1921, Val Accuracy: 92.48%
Epoch 50/50: 100%
Validation Epoch 50/50: 100%
Epoch 50/50 - Val Loss: 0.1878, Val Accuracy: 92.57%
=====
Training complete!
Best validation accuracy: 95.61%
=====
[OK] Loaded best model for metrics collection
[OK] Best model saved to 'resnet18_classifier_best.pth'
Collecting metrics on train set with best model...
```

## 5. Challenges & Learnings

- The biggest challenge was working with a very small labeled dataset and a large pool of unlabeled images.
- Many unlabeled images looked extremely similar, which made confidence-based labeling tricky at times.
- The model initially overfitted because the dataset was not clean, so improving labels became more important than tuning the model.
- Visualizing embeddings in 3LC helped clearly identify mislabeled samples and confusing clusters.
- A key learning was that data quality matters more than model complexity—small corrections created noticeable accuracy improvements.

## 6. Future Improvements

- Explore semi-supervised techniques like refined pseudo-labeling to improve confidence in difficult regions.
- Build a small manually curated validation set for more reliable accuracy tracking.
- Experiment with additional augmentation strategies for more robustness.
- Extend this workflow to multi-class problems or real-world datasets.
- Automate parts of the 3LC labeling workflow to speed up future iterations.

## 7. Conclusion

This project demonstrates a complete data-centric AI workflow powered by 3LC, showing how dataset refinement, not increased model complexity is the key driver of performance gains. Through iterative labeling, noise reduction, confidence analysis, clustering, and misclassification review, a robust dataset was created that enabled strong performance from a simple ResNet-18 classifier.