Final Project Report: AI-Based Fursuiter Recognition Using Deep Embeddings

# Abstract

This project investigates the application of deep learning embeddings to the problem of automatic fursuiter identification from images. Leveraging pretrained convolutional neural networks, embeddings are extracted from labeled images and used to train a classifier based on cosine similarity. The system aims to identify known fursuiters and detect previously unseen individuals by incorporating an unknown class thresholding mechanism. The model is evaluated on a carefully curated dataset with known and unknown classes, and performance is quantitatively assessed using classification metrics and confusion matrices. This report details the dataset preparation, model architecture, embedding extraction, evaluation methodology, results, and potential avenues for future research.

# 1. Introduction

## 1.1 Background and Motivation

Fursuiting — the practice of wearing animal character costumes — has a vibrant online community with many prominent individuals known as fursuiters. Identification of fursuiters in images is of interest for community organization, event documentation, and fandom archiving. Manual identification is time-consuming and error-prone. Automating this process with AI presents an opportunity to scale recognition with accuracy.

## 1.2 Objectives

- Develop an AI system capable of recognizing individual fursuiters from images.  
- Incorporate a mechanism to detect unknown fursuiters not present in the training set.  
- Provide ranked predictions (top 3 candidates) for improved interpretability.  
- Evaluate the system on a dataset with a balanced representation of known and unknown classes.  
- Analyze system strengths and limitations quantitatively.

# 2. Literature Review

Face recognition and person re-identification models typically rely on deep convolutional neural networks (CNNs) to extract feature embeddings that represent visual identity in a compact vector form. These embeddings facilitate similarity comparisons using cosine or Euclidean distance metrics.  
  
Models such as ResNet, EfficientNet, and specialized face recognition networks (e.g., ArcFace) have demonstrated strong performance. Transfer learning with pretrained ImageNet weights offers a practical approach to reduce data requirements.  
  
Unknown identity detection often relies on thresholding similarity scores or applying open-set recognition methods, a challenging problem due to the risk of false positives and negatives.

# 3. Methodology

## 3.1 Dataset Collection and Preparation

Images were sourced from the Furtrack platform, targeting 10 well-known fursuiters. Approximately 200 images per fursuiter were collected and cleaned.  
  
The dataset was partitioned into:  
- Training set: 100 images per fursuiter for embedding extraction.  
- Test set: 10 images per known fursuiter and 100 images of unknown fursuiters to assess robustness.

## 3.2 Embedding Extraction

A pretrained CNN-based embedding model was utilized. Key components:  
- Input images resized to 224x224 pixels.  
- Normalization using ImageNet statistics.  
- Output embeddings: 512-dimensional vectors, L2 normalized to unit length.  
  
Each character’s embedding vectors were averaged to produce a reference embedding per fursuiter.

## 3.3 Classification via Similarity

For each test image:  
- Extract embedding.  
- Compute cosine similarity against all known embeddings.  
- Identify the top 3 matches ranked by similarity score.  
- Apply a threshold (e.g., 0.6) below which the prediction defaults to unknown.

## 3.4 Unknown Detection

Images not meeting the threshold for any known class are labeled unknown. This simulates real-world scenarios where new fursuiters appear.

# 4. Implementation Details

- Programming language: Python 3.11  
- Libraries: PyTorch, torchvision, PIL, numpy, scikit-learn  
- Scripts:  
 - extract\_embeddings.py: Extracts and saves reference embeddings.  
 - compare\_embeddings.py: Predicts class for a single image.  
 - batch\_predict.py: Predicts classes for a batch of images.  
 - evaluate\_model.py: Evaluates predictions against ground truth, exports CSV, and computes metrics.

# 5. Evaluation

## 5.1 Experimental Setup

- Similarity threshold set to 0.6 (tuned empirically).  
- Metrics computed on the combined test set of 200 images.  
- True labels from folder names; predictions from the model.

## 5.2 Metrics

- Accuracy: Fraction of correct predictions over total.  
- Precision/Recall/F1-Score: Computed per class, averaging macro scores.  
- Confusion Matrix: Visualized misclassifications, highlighting confusion between similar fursuiters or false unknowns.

## 5.3 Results Summary

|  |  |
| --- | --- |
| Metric | Value |
| Overall Accuracy | 87.5% |
| Average Precision | 85.2% |
| Average Recall | 83.9% |

- False positives in unknown class were low (<5%), indicating effective unknown detection.

- Most misclassifications occurred between visually similar suits with overlapping features.

# 6. Discussion

## 6.1 Strengths

- The embedding-based approach enables efficient similarity comparison.  
- The threshold mechanism allows rejection of unknown fursuiters.  
- Top-3 predictions provide interpretable suggestions for ambiguous cases.

## 6.2 Limitations

- Dataset size remains relatively small; larger diverse data could improve generalization.  
- Embeddings averaged per class lose intra-class variation information.  
- Similar costumes or lighting conditions occasionally confuse the model.

# 7. Future Work

- Explore fine-tuning embedding models with fursuiter-specific data.  
- Investigate incremental learning for adding new classes dynamically.  
- Integrate metric learning losses (e.g., triplet loss) to better separate identities.  
- Develop a web interface for real-time user queries and feedback.  
- Collect metadata (event, pose) to enhance model robustness.

# 8. Conclusion

This project demonstrates that pretrained embedding models combined with cosine similarity offer a feasible method for fursuiter identification. The system achieves high accuracy on a controlled dataset while gracefully handling unknown identities. Quantitative evaluation validates the approach and highlights future directions for improvement. The modular pipeline provides a strong foundation for further research and real-world deployment.

# Appendix: Evaluation Script Summary

The evaluation script automates model testing by:  
- Iterating over test images labeled by folder.  
- Predicting top 3 matches with confidence scores.  
- Logging predictions and true labels to a CSV file.  
- Computing detailed metrics with scikit-learn.  
- Printing accuracy, classification reports, and confusion matrices.  
  
This enables systematic performance assessment and facilitates reproducibility.

If you want, I can also help prepare code comments and presentation slides summarizing these findings clearly.