National Sun Yat-sen University
Department of Information Management

SpOrtsNet：A Transfer Learning Approach for Olympic Sports Recognition

Authors : B104020006莊子儀、B104020008林冠儒、B104020037董昱辰

JAN 3, 2025

Deep Learning Final Report

1. Cover page

- **Topic:** SpOrtsNet：A Transfer Learning Approach for Olympic Sports Recognition

- **Authors:** B104020006莊子儀、B104020008林冠儒、B104020037董昱辰

- **Research Problem**

This study addresses the challenge of classifying Olympic sports imagery. Traditional image classification models face two critical challenges when processing sports images: first, the high similarity and variability in athletic movements, and second, the complex variations in lighting, angles, and backgrounds in real-world applications. These challenges become particularly significant after expanding to 15 sports categories, making feature discrimination between classes more demanding.

- **Solution Approach**

We implemented a two-phase strategy to address these challenges:

1. Phase One involved a comprehensive model evaluation, comparing various architectures including custom MLP, ResNet18, ShuffleNet, EfficientNet, and MobileNet, while testing different optimizers (SGD, Adam, AdamW) for performance optimization.
2. Phase Two employed a progressive optimization strategy:
   - Implementation of advanced data augmentation techniques including Cutout and Color Jittering
   - Integration of Contrastive Learning to enhance feature representation capabilities
   - Application of Prototype Learning to strengthen class feature discrimination

- **Most Interesting Findings**

1. Significance of Architecture Selection: The substantial performance gap between custom MLP (24%) and EfficientNet (91.91%) demonstrates the crucial role of pre-trained models in sports image classification. As shown in our previous experiment in Figure 1,the training curves demonstrate stable convergence and consistent performance of the EfficientNet model.

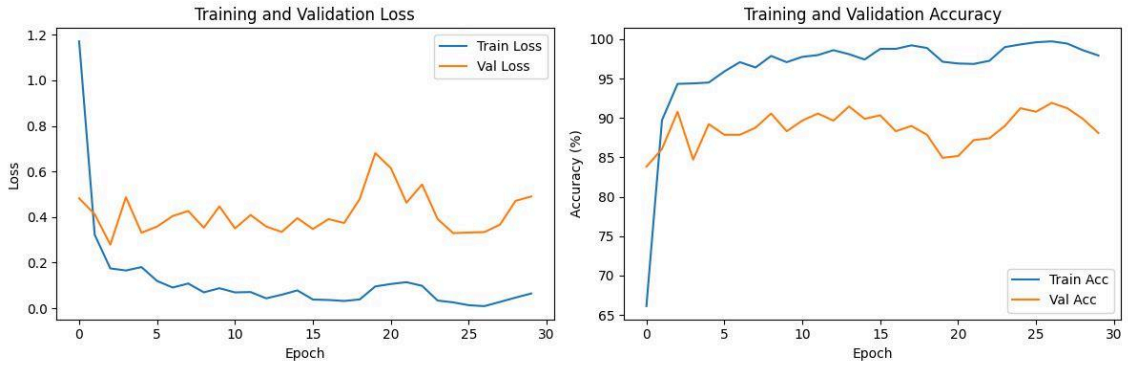


Figure 1 Training curves for direct fine-tuning approach: (a) Loss curves (b) Accuracy curves

2. Optimizer Impact: The same EfficientNet architecture exhibited significant performance variations with different optimizers, with Adam/AdamW showing notably better convergence characteristics compared to SGD.
3. Model Misclassification Patterns Reflecting Human Visual Cognition : As illustrated in Figure 2(a) and 2(b), the model's predictions exhibit both successful and erroneous classifications across various Olympic sports. The misclassification patterns emerge in three primary categories:

    - In sports with similar motion patterns, such as basketball being misclassified as handball, revealing the model's capture of common features in ball-handling movements
    - In sports with similar venue characteristics, such as rugby being misclassified as football, highlighting the influence of environmental context on model decisions
    - In cases of similar postures, such as the mutual misclassification between breaking, table tennis, demonstrating the model's interpretation of instantaneous motion poses

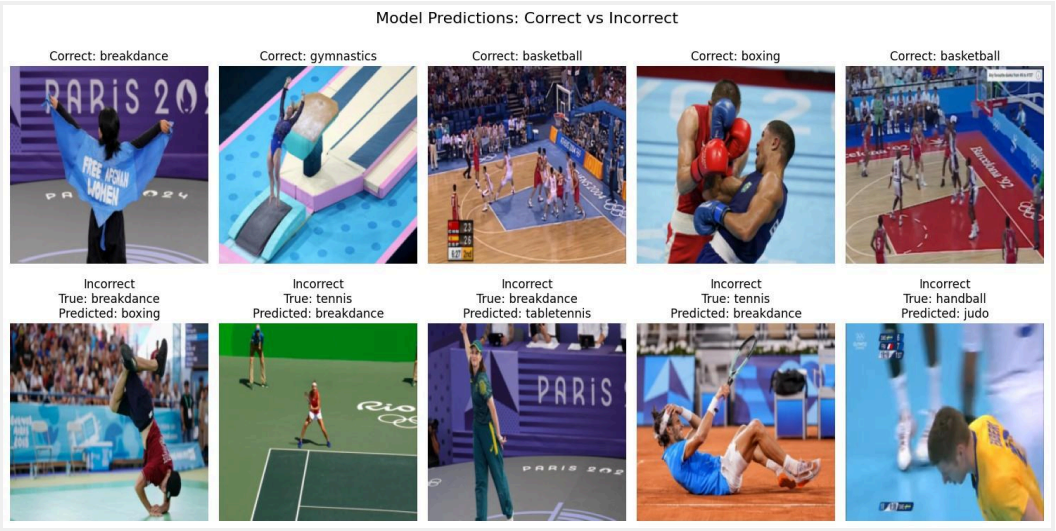


Figure 2 Visual examples of model predictions: (a) First set of correct and incorrect predictions (b) Second set of correct and incorrect predictions

2. Introduction

- **Problem Definition & Motivation**

This research focuses on the challenges of Olympic sports image classification, which presents unique technical complexities in computer vision. The study primarily addresses two key technical issues:

1. Similarity and Variability of Athletic Movements
   - Overlapping Basic Movements Across Sports: For example, basketball and handball both involve throwing motions, while tennis and table tennis share swinging movements.
   - Movement Diversity Within Individual Sports: A single sport may include various action phases such as running, jumping, and throwing.
   - Subtle Feature Differences: Models need to identify and distinguish crucial differences between similar movements.
2. Environmental Complexity in Real-world Scenarios
   - Lighting Condition Variations: Different light intensities and directions in indoor and outdoor venues.
   - Diversity of Camera Angles: Various shooting angles and distances.
   - Venue Background Variations: Visual feature changes of similar venues (such as courts) under different environments.

These challenges become more pronounced as the research scope expands to 15 sports categories. The significance of this research lies not only in improving the accuracy of sports image classification but also in establishing theoretical foundations for practical applications such as automated content categorization, sports performance analysis, and educational training. Therefore, developing more precise feature identification techniques is crucial for enhancing the practical applicability of sports image classification.

- **Background Materials & Related Work**

Based on the result graph last time, we found out that there is a gap between training and validation accuracy, which means that the model is overfit. To solve this problem, we implemented **dropout** and **added 50 more pictures to each category**.

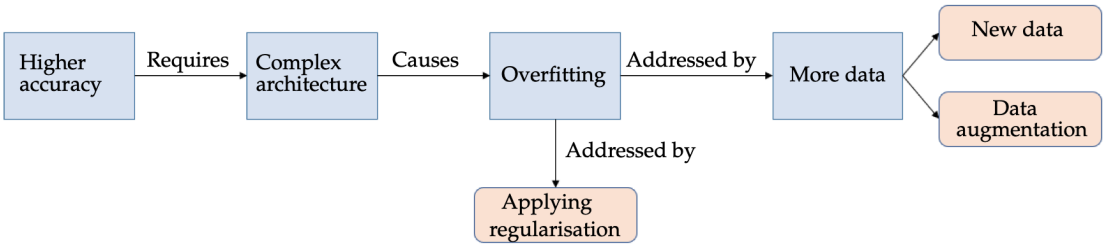


Figure 3 Flowchart of model overfitting problem and solutions.
(From: Data Augmentation in Classification and Segmentation: A Survey and New Strategies)

- **Data Augmentation in Image Classification**

Aside from dropout and increasing the dataset size, we also implemented data augmentation techniques such as **image rotation, color jittering and cutout** to improve our model. Rotation helped the model better manage a variety of camera angles commonly found in real-world sports situations. Color jittering and cutout helped the model focus on essential features by adjusting the brightness or contrast of the image and by randomly masking parts of the image respectively.

- **Contrastive Learning in Image Classification**

Contrastive learning has emerged as a significant advancement in self-supervised visual representation learning. The fundamental principle involves training models to learn discriminative features by comparing similar and dissimilar image pairs through a CNN-MLP architecture. As illustrated in the Figure4, the model processes augmented versions of images through CNN encoders and MLP projection heads, optimizing the representation space to minimize distance between positive pairs while maximizing it between negative pairs.

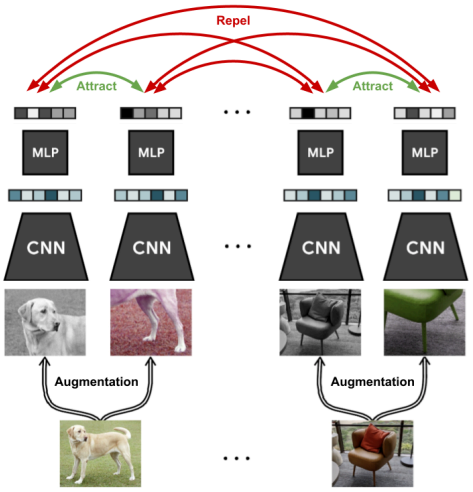


Figure 4 Illustration of contrastive learning framework with CNN-MLP architecture and positive/negative pair optimization.

Recent developments include Dense Contrastive Learning (DenseCL), which enhances feature extraction capabilities using a ResNet backbone for dense prediction tasks. Building upon these findings, our research adapts the contrastive learning framework by implementing EfficientNet as the backbone architecture. This choice is motivated by EfficientNet's demonstrated efficiency in model scaling and superior performance in image classification tasks with limited data availability.

The effectiveness of supervised contrastive learning (SCL) is particularly relevant for our Olympic sports classification task, where label information guides the creation of meaningful positive and negative pairs. This supervision, combined with strategic data augmentation, enables the model to learn robust feature representations that are both invariant to specific augmentations and discriminative between different sports categories, even with a constrained dataset size.

- **Prototype-Based Learning in Image Classification**

Prototype-based learning in image classification marks a notable advancement over traditional deep learning methods by embedding interpretable, class-representative examples directly into the learning framework. When paired with EfficientNet's architecture, this approach not only enhances interpretability but also leverages computational efficiency to improve classification performance.

Prototypes act as learned templates that encapsulate key characteristics of each class, akin to how humans identify objects by referencing stored mental representations. Research by Li et al. (2018) highlights that prototype-based learning fosters more transparent decision-making processes. Each prototype functions as a tangible, visualizable example, allowing the model's predictions to be more easily understood and analyzed.

This methodology proves particularly valuable in the context of Olympic sports classification, where fine-grained distinctions such as differentiating swimming styles or similar gymnastic movements pose significant challenges. By utilizing prototype-based learning, EfficientNet can achieve more precise and interpretable decision-making while preserving its computational advantages, making it an ideal solution for real-world applications in sports analysis and automated event classification.

- **Project Approach Summary**
  - **Data augmentation (Rotation、Color Jittering、Cutout)**

    Implementing data augmentation methods can improve the performance of the model and also solve the problem of overfitting. There are various methods of data augmentation, in this project, we decided to implement rotation, color jittering and cutout, which are some of the popular methods.

  - **Contrastive learning**

    Building upon the demonstrated effectiveness of contrastive learning in self-supervised representation learning and its success with limited data, we implement this approach for Olympic sports image classification. Our methodology employs contrastive learning for initial feature extraction, followed by supervised classification. The architecture utilizes EfficientNet-B0 as the backbone network, complemented by projection and classification heads, aiming to develop discriminative features despite dataset constraints.

  - **Prototype-based learning**

    Integrating a prototype layer with EfficientNet-B0 for Olympic sports classification, our architecture processes images through feature extraction, compares them with learned class prototypes, and performs classification based on prototype distances to enable interpretable predictions.

- **Project Novelty**

Our project offers novel contributions to Olympic sports image classification through a comprehensive comparative study of multiple advanced techniques. While existing literature predominantly utilizes ResNet for pre-training and fine-tuning [Mohamad et al., 2020], our extensive experimentation identifies EfficientNet as the optimal backbone architecture for this specific task. Furthermore, we conduct comparative experiments on three distinct approaches: enhanced data augmentation techniques (including rotation, color jitter, and cutout), contrastive learning, and prototype learning. Unlike previous studies that typically focus on single methodologies, such as transfer learning with data augmentation [Mohamad et al., 2020], our research evaluates and compares the effectiveness of these different approaches, providing comprehensive insights into their relative performance in sports image classification under limited data scenarios.

3. Details of the approach

- **Approach1 : Data Augmentation**
  - **Technical Components**

This approach focuses on the model's generalization by applying data augmentation techniques (Image Rotation, Color Jittering, Cutout). These methods are used to solve the overfitting of our model. Key technical components include:

1. **Data Augmentation Module:** Enriches the dataset for robust training by applying different techniques.
2. **Feature Extractor (EfficientNet-B0)**: Pre-trained model which performed the best.
3. **Classification Head**: Dropout and batch normalization are used for regularization in fully connected layers for final classification.

- **System Architecture**

  The system architecture consists of two primary components:
  1. Preprocessing Module: Augments original images with the techniques below:
     a. **Image Rotation**: Applies random rotations within a ±30° range
     b. **Color Jittering**: Since the color theme of different sports are different, we used color jittering to prevent the model from relying on colors solely for classification.
     c. **Cutout**: Randomly masks parts if the image
  2. Feature Extraction and Classification:
     a. EfficientNet-B0 backbone
     b. A multi-layer classification head that predicts the sports category

- **Implementation Details**
  1. **Backbone Model and model classifier:** To address the overfitting problem, we also implemented dropout, and found out that the model works well with the rate 0.3.

```
def create_model(num_classes, dropout_rate=0.3):
    # 載入預訓練的 EfficientNet-B0 模型
    model = models.efficientnet_b0(weights='DEFAULT')


    model.classifier[1] = nn.Sequential(
        nn.Dropout(dropout_rate),  # 添加 Dropout 層
        nn.Linear(model.classifier[1].in_features, num_classes)
    )

    return model.to(DEVICE)
```

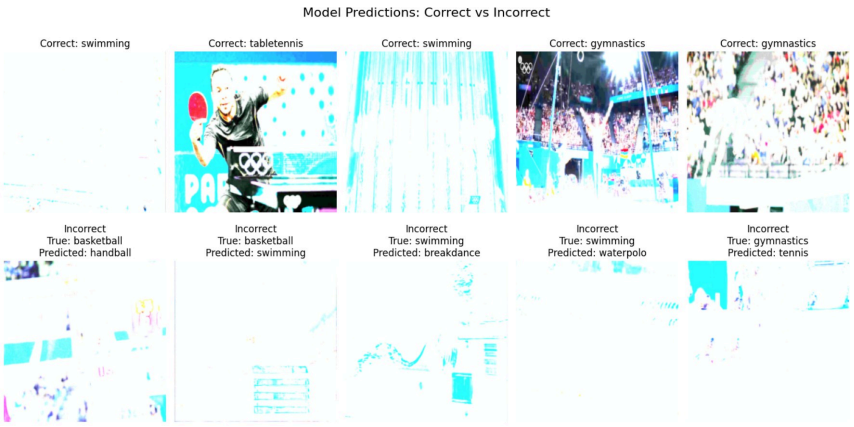
  2. **Data Augmentation Implementation**:

Rotation:

```
data_transforms = transforms.Compose([
   transforms.RandomHorizontalFlip(),
   transforms.RandomRotation(30),
   transforms.Resize((224, 224)),
   transforms.ToTensor(),
      transforms.Normalize(mean=[0.485,  0.456,  0.406],  std=[0.229,
   0.224, 0.225])
])
```

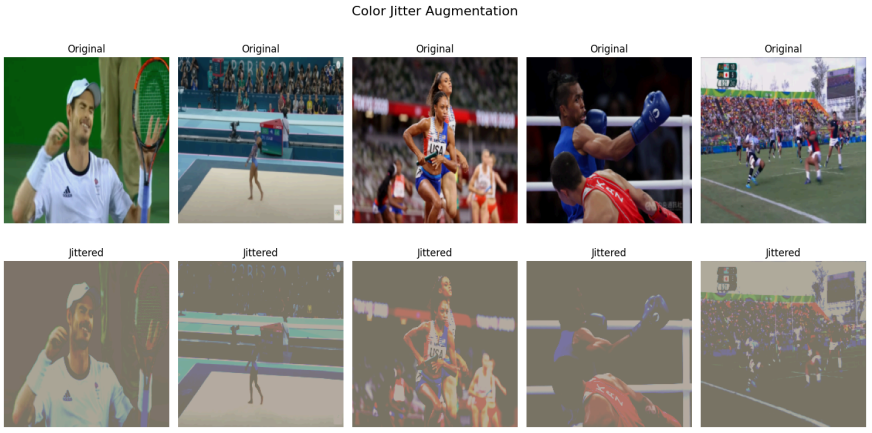
Color Jittering:

At first, we did not adjust the parameters for color jittering correctly, which results in poor model performance, with validation accuracy around 75% only. Based on this investigation, we found out that without proper adjustments, the images will have excessive exposure (the image results are shown below), negatively and seriously impacting the model's ability to predict the category of our images.



After adjusting the parameters, the model performance is better.

```
data_transforms = transforms.Compose([
    transforms.ColorJitter( #adjust parameters
        brightness=0.1,
        contrast=0.1,
        saturation=0.1,
        hue=0.02
    ),
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
      transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229,
  0.224, 0.225])
])
```



Cutout:

```
class Cutout:
    def __init__(self, size):
        self.size = size

    def __call__(self, img):
        c, h, w = img.shape
        x = random.randint(0, w - 1)
        y = random.randint(0, h - 1)
        x1 = max(0, x - self.size // 2)
        x2 = min(w, x + self.size // 2)
        y1 = max(0, y - self.size // 2)
        y2 = min(h, y + self.size // 2)
        img[:, y1:y2, x1:x2] = 0
        return img
```

- **Approach2 : Contrastive Learning**
  - **Model Architecture Overview**

This study adopts a training strategy that combines contrastive learning followed by classification. The process begins with contrastive learning to obtain general image feature representations, then utilizes the pre-trained model as a feature extractor with frozen parameters to establish the classification model.

  - **System Architecture**

The system employs a dual-path architecture for processing paired image inputs, implementing feature learning through a shared weight mechanism. This design enables the model to learn discriminative features by comparing similar and dissimilar image pairs. The system employs a dual-path architecture with the following key components.

  - **Technical Components**

1. **Feature Extractor** (Backbone: EfficientNet-B0)

A pre-trained backbone network (1280-dimensional output) that extracts high-dimensional visual features while maintaining computational efficiency.

```
encoder = torchvision.models.efficientnet_b0(pretrained=True)
encoder.classifier = nn.Identity()  # Remove classification layer
feature_dim = 1280  # Output dimension
```

2. **Projection Head** (Contrastive Learning Phase)

A two-layer structure for feature transformation:

```
projector = nn.Sequential(
    nn.Linear(1280, 1280),  # Maintain dimension
    nn.ReLU(),         # Non-linear activation
    nn.Linear(1280, 128)  # Dimension reduction
)
```

3. **Classification Head** (Classification Phase)

Built upon the frozen feature extractor:

```
self.classifier = nn.Sequential(
    nn.Linear(1280, 256),       # Dimension reduction
    nn.BatchNorm1d(256),        # Normalization layer
    nn.ReLU(inplace=True),      # Non-linear activation
    nn.Dropout(0.3),            # Regularization
    nn.Linear(256, num_classes)
)
```

4. **Contrastive Loss** (InfoNCE)

Loss function that maximizes agreement between positive pairs while minimizing similarity between negative pairs in the embedding space.

The InfoNCE loss is computed as :

$$\mathcal{L} = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(sim(z_i, z_k)/\tau)}$$

- z_i, z_j are encoded representations
- $sim(u, v)$ is cosine similarity
- $\tau = 0.07$ is temperature parameter
- N is batch size

- **Training Process**

1. Data Preprocessing
   - Image resizing to 224x224
   - Color jittering (brightness=0.8, contrast=0.8, saturation=0.8, hue=0.2)
   - Random horizontal flipping
   - Random grayscale conversion (p=0.2)
   - Image normalization (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

2. Training Configuration
   - Batch size: 32
   - Training epochs: 20
   - Optimizer: Adam with learning rate 0.001
   - Learning rate decay: 10% per 10 epochs
   - Feature vector L2 normalization

- **Approach 3: Prototype-Based Learning**
- **Model Architecture Overview**

The model implements a prototype-based learning strategy where class prototypes serve as learnable parameters, and classification is performed based on the distance between input features and these prototypes. The architecture consists of a feature extractor followed by prototype-based classification.

- **System Architecture**

The system utilizes a single-path architecture with EfficientNet-B0 as the base model, followed by prototype-based classification. The key innovation is replacing traditional softmax classification with prototype learning, where each class is represented by a prototype vector in the feature space.

- **Technical Components**

1. **Feature Extractor**: Use EfficientNet-B0 as feature extractor and outputs 1280-dimensional feature vectors.

```
class ProtypicalModel(nn.Module):
    def __init__(self, base_model, num_classes):
        super(PrototypicalModel, self).__init__()
        self.base_model = base_model
        self.num_classes = num_classes
        self.prototypes = nn.Parameter(torch.zeros(num_classes, 1280))
```

2. **Distance-based Classification**: The model computes distances between extracted features and class prototypes.

```
    def forward(self, x):
        features = self.base_model.features(x)
        features = self.base_model.avgpool(features)
        features = features.flatten(1)
        dists = torch.cdist(features, self.prototypes)
        logits = -dists
        return logits
```

3. **Loss Function**: Uses cross-entropy loss on the negative distances.

```
criterion = nn.CrossEntropyLoss()
```

- **Training Process**

1. Data Preprocessing
   - Image resizing to 224x224
   - Standard normalization with ImageNet statistics(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
   - Data augmentation techniques

2. Training Configuration
   - Batch size: 32
   - Training epochs: 30
   - Optimizer: Adam
   - Learning rate: 0.001

4. Results

- **External Resources**

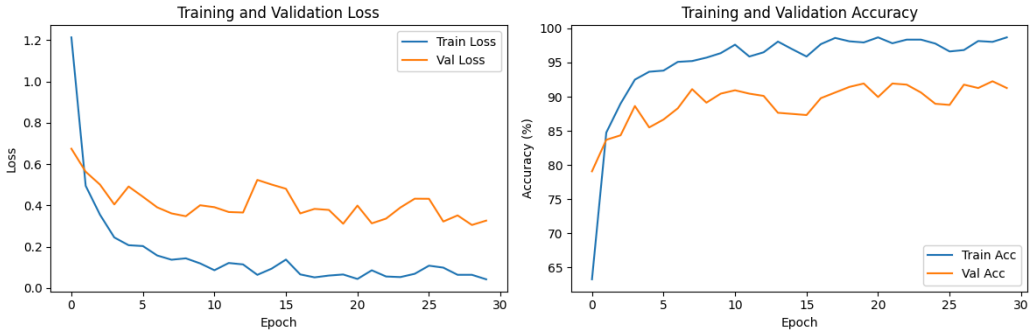
  The experimental environment configuration is as follows:

  1. Hardware Environment : Training is conducted using Google Colab T4 GPU, with each training epoch taking 150 seconds.
  2. Pre-trained Resources : Using EfficientNet-B0 backbone network and ImageNet pre-trained weights.

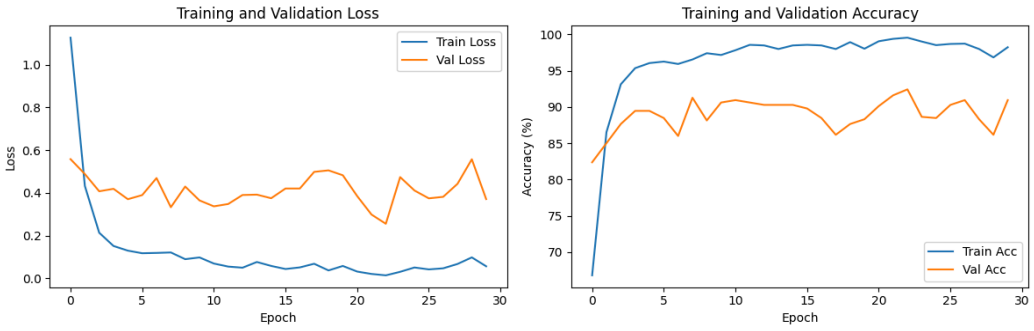
- **Approach 1 : Data Augmentation Result**

- **Quantitative Evaluation**

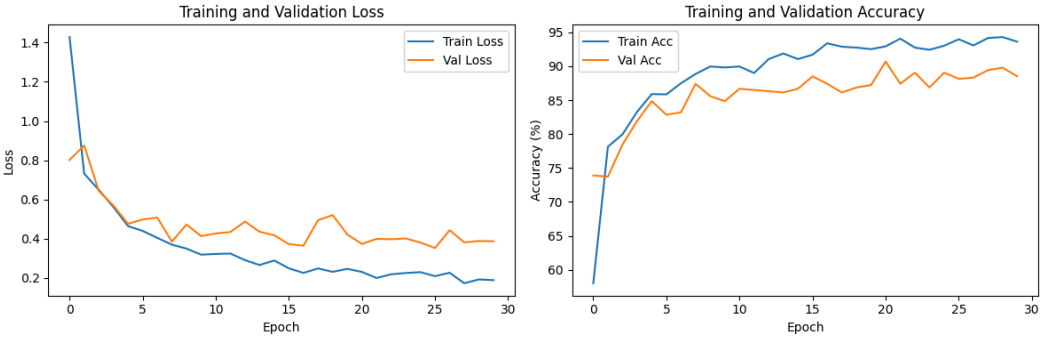
  A. **Image Rotation:** Accuracy: 92.26%, Training Time: 57min14sec



  B. **Color Jittering:** Accuracy: 92.42%, Training Time: 3hr40min29sec



  C. **Cutout:** Accuracy: 90.69%, Training Time: 37min51s **(As we can see, the gap between training accuracy and validation accuracy has declined.)**



- **Approach 2 : Contrastive Learning Result**

- **Experimental Protocols**

  This experiment consists of two phases: contrastive learning pre-training and classification model training.

  1. Contrastive Learning Pre-training Phase

The model establishes feature representation through contrastive learning method. Due to computational resource limitations, training epochs are set to 20. Using InfoNCE loss function for training, the loss value decreases from 0.2 to below 0.02, with each training epoch taking approximately 150 seconds.

2. Classification Training Phase

The classification model undergoes 20 training epochs using cross-entropy loss function. The dataset is split into training and validation sets at an 8:2 ratio.

- **Quantitative Evaluation**

Model evaluation is based on training process metrics.

1. Contrastive Learning Phase

As shown in Figure 5(a), the model shows significant loss value reduction in the first 5 training epochs, converging to 0.005. Figure 5(b) demonstrates the training time per epoch, initially starting at 1000 seconds before stabilizing at approximately 150 seconds per epoch.

2. Classification Model Performance Metrics:

As illustrated in Figure 2, the model achieves the following performance:

- Best Training Accuracy: 99.08%
- Best Validation Accuracy: 80.50%
- Final Training Loss: 0.0450
- Final Validation Loss: 0.7352
- Total Training Time: 1751.18 seconds
- Average Epoch Time: 87.56 seconds

3. Model Convergence Analysis

The training process, shown in Figure 6(a), demonstrates the loss curves for both training and validation sets. The training loss decreases from 1.6 to 0.0450, while validation loss stabilizes at 0.7352. Figure 6(b) shows the accuracy curves, where the 20% gap between training (99.08%) and validation accuracy (80.50%) indicates moderate overfitting, though the model maintains generalization capability.

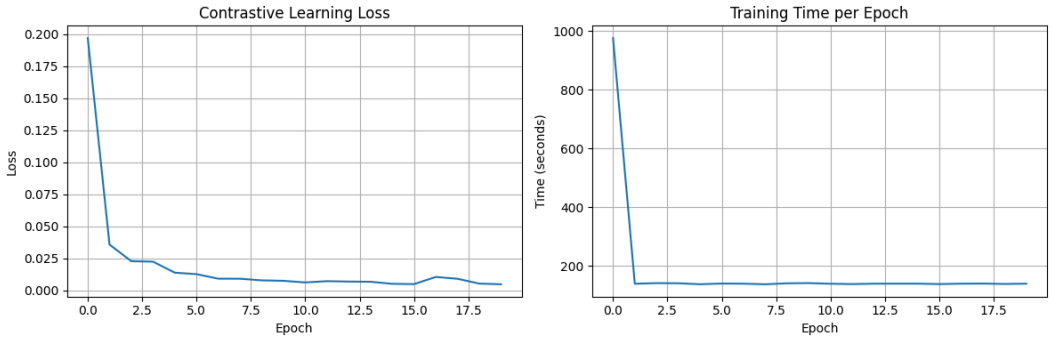


Figure 5 (a) Contrastive Learning Loss over epochs (b) Training Time per Epoch

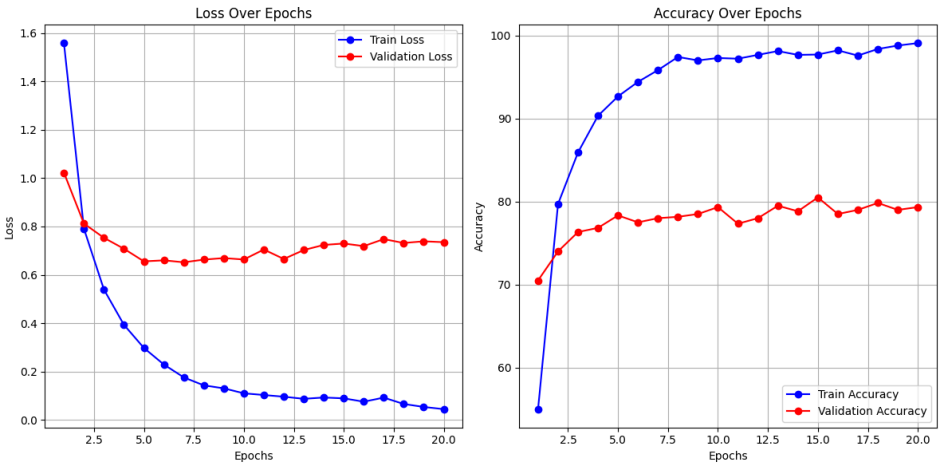


Figure 6 (a) Loss Over Epochs for classification (b) Accuracy Over Epochs for classification

- **Approach 3: Prototype-Based Learning Result**
  - **Experimental Protocols**

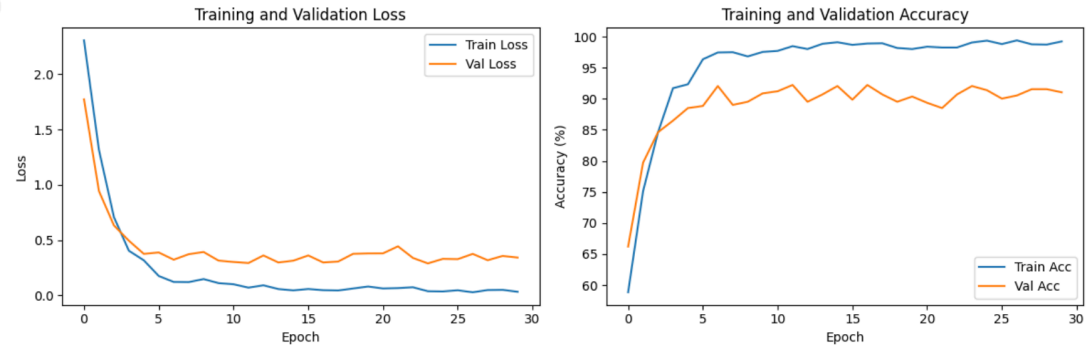
    We employ EfficientNet-B0 as the base model with prototypical extensions for sports image classification. The dataset is split into 80% for training and 20% for validation. Training configuration includes a batch size of 32, running for 30 epochs with a learning rate of 0.001, targeting 15 sports categories using cross-entropy loss.

  - **Quantitative Evaluation**

    The learning curves demonstrate excellent training stability, with training loss declining from 2.3072 to 0.0321 and validation loss decreasing from 1.7737 to 0.3416. This parallel reduction indicates strong generalization without significant overfitting.

    The model achieves a best training accuracy of 99.41% and peak validation accuracy of 92.23%. The total training completes in 3165 seconds (52 minutes 45 seconds), averaging 87 seconds per epoch.

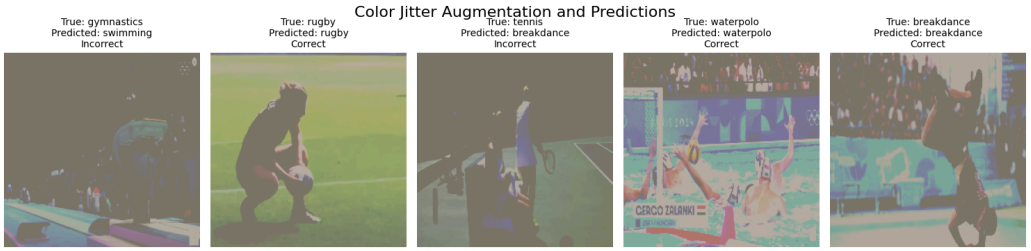
    The convergence shows rapid early improvement with validation accuracy exceeding 88% within 5 epochs. The best validation accuracy of 92.23% is achieved at epoch 12, maintaining stable performance around 90% afterward.

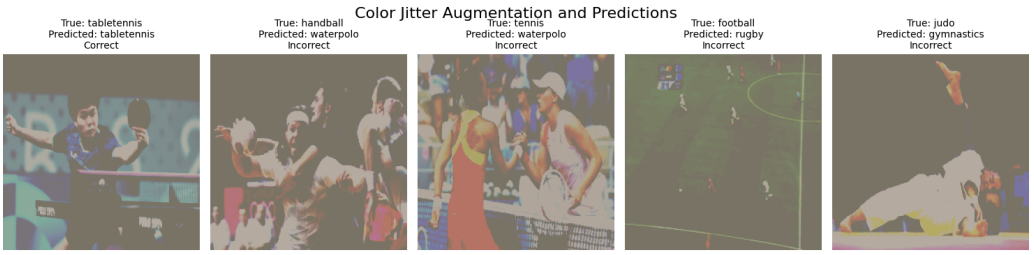


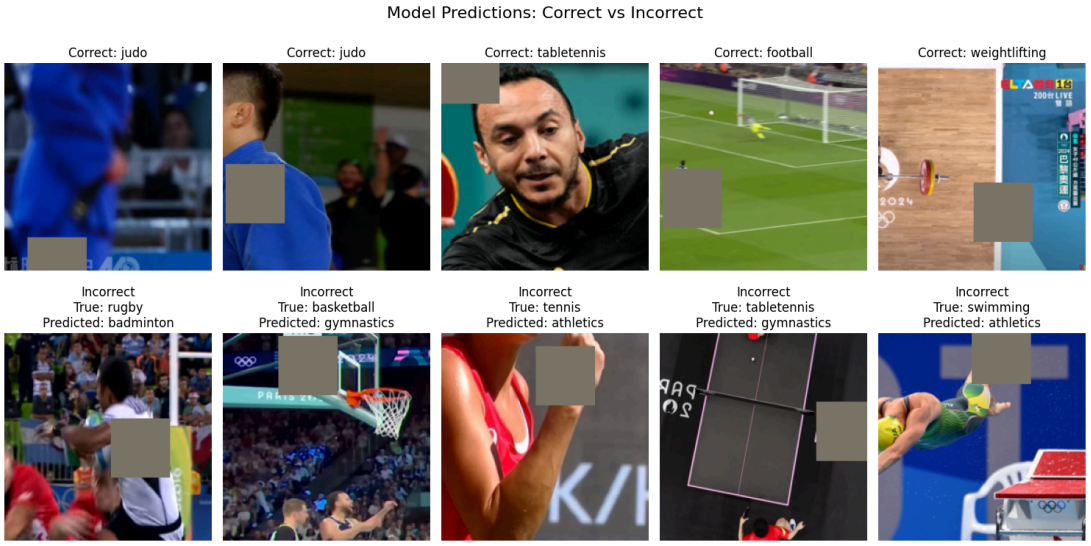
5. Discussion and conclusions

- **Insights from Data Augmentation**

  From the predicted results, we discovered something interesting. For football, the model was able to accurately predict the class if the goal was visible. During image rotation and color jittering, the model was unable to predict football correctly because the goal was absent. In the cutout technique, the masked area did not cover the important feature (the goal), so the model made an accurate prediction. For water polo, the critical feature is the player's hat. The model incorrectly identified a tennis image as water polo during color jittering. We think this happened as a result of the model mistaking the tennis player's headgear as a water polo hat.







- **Insights from Prototype-based Learning**

  Our implementation of prototype-based learning with EfficientNet-B0 achieved a validation accuracy of 92.23%, showing a modest improvement of 0.32 percentage points over the baseline model's 91.91%. The learning curves demonstrated rapid convergence and notable stability throughout training, suggesting that prototype-based learning contributes to more efficient training dynamics. While the accuracy gain is incremental, the approach's stability during training makes it a promising modification to traditional CNN architectures that warrants further investigation with larger datasets and more sophisticated augmentation techniques.

- **Conclusion**

  Although data augmentation and prototype-based learning do slightly solve the problem of overfitting and enhances the accuracy, the accuracy in the contrastive learning does not improve. In addition, the overfitting problem became more serious in contrastive learning. We believe if we add more photos to our dataset, the problem might be solved.

- **Future Improvements**

  In this project, we found out that the greatest limitation was the GPU limit in GoogleColab. After we enhanced the dataset to 3000 pictures, it took lots of time to run, and we can only run 2-3 times a day, sometimes it will even be interrupted due to the long running time. Hence, in the future, if applicable, we would want to enhance the functionality of our GPU in order to train the model more efficiently.

  In addition, for data augmentation, there are several other methods that we did not apply in this project. If we have the time and resources, we plan to apply different methods, such as Grid Mask, Mixup, CutMix and also some advanced approaches such as GAN.

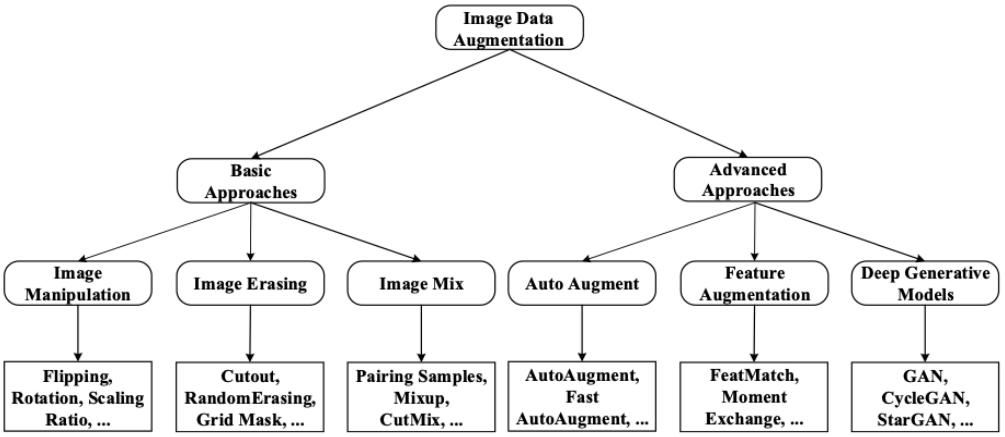


Figure 7 Taxonomy of image data augmentation methods.

- **Negative Results (if applicable)**
  - **Challenges of Contrastive Learning in Image Classification**

    Contrastive learning for pretraining did not outperform direct transfer learning in image classification tasks, likely due to several factors. The learned representations may lack the specificity needed for classification, particularly when the pretraining and downstream tasks differ significantly. Additionally, the limited computational resources in Colab and the simplicity of the model architecture could restrict the effectiveness of contrastive learning. Suboptimal hyperparameter configurations, such as batch size and temperature, may further contribute to performance limitations.

    To address these issues, future work should prioritize enhancing computational resources and increasing batch size to provide more diverse negative samples. Adopting more advanced architectures, such as combining contrastive learning with adaptive fine-tuning, could improve representation transfer. Additionally, robust data augmentation and task-specific hyperparameter tuning are crucial for maximizing the potential of contrastive learning in downstream applications.

6. Statement of individual contribution

- **B104020006 莊子儀**
  - Proposal Content Writing
  - Image Data Collection
  - Progress Update (MLP, Transfer Learning (ResNet18, ShuffleNet) Implementation)
  - Final Report: Literature Review, Writing, and Implementation of Contrastive Learning

- **B104020008 林冠儒**
  - Image Data Collection
  - Progress Update (MLP, Transfer Learning (MobileNet) Implementation)
  - Final Report: Literature Review, Writing, and Implementation of Prototype-Based Learning

- **B104020037 董昱辰**
  - Image Data Collection, Data Cleaning
  - Proposal Content Writing
  - Progress Update (MLP, Transfer Learning (EfficientNet) Implementation)
  - Final Report: Literature Review, Writing, and Implementation of Data Augmentation

7. Reference

- Kaggle - OGED-Olympic Games Event Dateset
- Olympic Games Event Recognition via Transfer Learning with Photobombing Guided Data Augmentation
- [2011.09157] Dense Contrastive Learning for Self-Supervised Visual Pre-Training
- [2004.11362] Supervised Contrastive Learning
- Deep Learning for Case-Based Reasoning Through Prototypes: A Neural Network That Explains Its Predictions | Proceedings of the AAAI Conference on Artificial Intelligence