Deep Clustering for Unsupervised Learning of Visual Features

E4040.2024Fall.SZQA.zs2699.yq2411.jz3849
Zhuoya Shi zs2699, Yuchen Qiu yq2411, Jieyuhan Zhu jz3849
Columbia University

Abstract

Convolutional Neural Networks (CNNs) rely on high-cost labeled datasets, which are prone to introducing biases and limiting their applicability under limited data conditions. As a result, unsupervised learning has garnered significant attention. Caron et al. proposed the DeepCluster model, a clustering-based unsupervised learning method that alternates between feature clustering and network training, enabling high-quality feature learning without labeled data. This method demonstrated outstanding performance on large-scale datasets, significantly narrowing the performance gap between supervised and unsupervised learning. This project aims to replicate this DeepCluster model to validate its applicability and performance under limited computational resources. By conducting small-scale experiments, the project utilizes the CIFAR-10 dataset as a substitute for large-scale datasets and adopts lightweight architectures, such as MobileNet and a custom shallow CNN, to address the challenges of small-scale data and resource-constrained environments. Through input dimension optimization and enhanced model generalization, the project effectively overcomes resource limitations and validates the feasibility of the DeepCluster framework on small-scale datasets and constrained computing environments. The project successfully achieves its initial goals while providing valuable practical experience for research in deep learning and unsupervised learning methodologies.

1. Introduction

CNN has become a key component in most computer vision applications [1]. Their properties enable them to improve the generalization ability of models under limited data. However, current pre-training methods usually rely on labeled datasets, whose construction demands significant time and cost [2]. Moreover, attempts to replace manual annotations with metadata can introduce unpredictable biases [3], further limiting the generalization ability and applicability of the model.

Therefore, unsupervised learning has gradually become a highly attractive research direction. Compared with traditional supervised learning, unsupervised learning does not rely on manual labels, especially in areas where annotations are scarce (such as medical images and satellite images). However, most existing unsupervised methods are based on fixed features or linear models, which are challenging to integrate seamlessly with deep learning frameworks, thereby limiting their effectiveness in large-scale end-to-end training [4].

In 2019, Caron et al. proposed the DeepCluster model, an innovative clustering-based unsupervised deep learning method capable of achieving end-to-end large-scale training of CNN [5]. The core mechanism of DeepCluster involves alternating between feature clustering and network training to optimize feature representations. It uses a CNN network to extract data features, applies k-means clustering to generate pseudo-labels, and updates network weights based on these pseudo-labels. Through this iterative process, DeepCluster learns high-quality feature representations without requiring manual annotations, significantly reducing reliance on labeled data. This method demonstrated exceptional performance on large-scale datasets, outperforming existing unsupervised methods in classification, detection, and transfer learning tasks, and narrowing the performance gap between unsupervised learning and supervised learning. The innovation and broad applicability of the DeepCluster model offer a new approach to unsupervised visual feature learning and lay a solid foundation for building efficient and robust deep learning models.

To achieve this goal, we aim to replicate the DeepCluster framework to verify its applicability and performance through an in-depth study of its core mechanism. Due to the limitation of computing resources, this project simplified the DeepCluster framework and conducted small-scale experiments. Additionally, through this reproduction process, we can not only deepen our understanding of the fields of deep learning and unsupervised learning, but also

accumulate practical experience, laying a solid foundation for subsequent research and application.

The main technical challenge of this project is that the computing resources are limited, which makes it difficult for us to fully reproduce the method in the paper using large-scale datasets (such as ImageNet [6] and YFCC100M [7]) and complex network architectures (such as AlexNet [8] and VGG16 [9]). Therefore, we chose to replace the original large-scale dataset with CIFAR-10 [10] to complete training and clustering under limited resources. However, CIFAR-10's small scale and low resolution pose challenges for ensuring stability and convergence during the alternating training process. To address these challenges, this project incorporates multiple adaptations and optimizations to the framework: replacing AlexNet and VGG16 with the lightweight MobileNet [11] and a custom shallow CNN to reduce model complexity while enabling performance comparison; reconstructing the DeepCluster process using TensorFlow/Keras [12] to optimize resource utilization; and applying grayscale conversion and data augmentation to reduce input dimensions and enhance the model's generalization ability. These strategies effectively tackle resource constraints and demonstrate the applicability of the DeepCluster framework in small-scale datasets and resource-limited computing environments.

2. Summary of the Original Paper

2.1 Methodology of the Original Paper

The DeepCluster proposed in the original paper is an unsupervised deep learning method based on clustering. Its core idea is to generate pseudo-labels through clustering and iteratively optimize the training process in combination with a CNN. Specifically, the method first uses a CNN to extract image features, then applies k-means clustering to generate pseudo-labels, and subsequently uses these pseudo-labels for supervised training of the network to optimize feature representations. This iterative process continuously refines the quality of the feature space. The original paper validated the effectiveness of this method on large-scale datasets, demonstrating its outstanding performance in classification and transfer learning tasks.

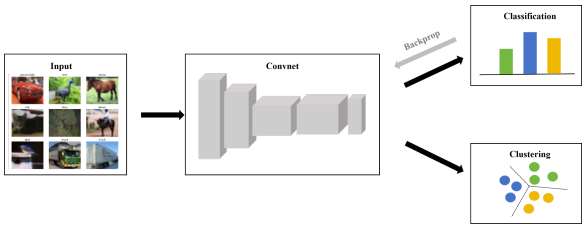

Fig.1: DeepCluster Workflow Chart

2.2 Key Results of the Original Paper

The DeepCluster proposed in the original paper demonstrated an outstanding performance in unsupervised learning tasks.On the Pascal VOC transfer tasks, when pre-trained on ImageNet, DeepCluster achieved classification accuracy of 72.0% (fc6-8) and 73.7% (all layers), detection accuracy of 51.4% (fc6-8) and 55.4% (all layers), and segmentation accuracy of 43.2% (fc6-8) and 45.1% (all layers), outperforming previous methods by +9.7%, +8.2%, and +9.3%, respectively.

Compared to other unsupervised learning methods, the DeepCluster showed consistent superiority. It improved detection accuracy by +4.3% over Doersch et al. on AlexNet and by +2.7% over Wang et al. on VGG-16 [13]. For instance-level image retrieval, DeepCluster achieved a mean Average Precision (mAP) of 61.0% on the Oxford Buildings dataset (+25.6% over Doersch et al.) and 72.0% on the Paris dataset (+18.9%) [14].

Additionally, the robustness of DeepCluster on training data was validated when pre-trained on an imbalanced subset. It still improved classification and segmentation accuracy by +4.3% and +4.5%. The DeepCluster also further narrows the performance gap with supervised methods to only 1.4% under deeper architectures such as VGG-16, and can generate general visual features that are applicable to a variety of tasks without relying on extensive domain knowledge.

Based on these key results, the DeepCluster proposed in the original paper, provides a scalable and domain-independent unsupervised deep learning method, which performs well in tasks such as classification, detection, and instance retrieval. It outperforms most existing unsupervised methods and significantly narrows the performance gap between supervised and unsupervised models, validating its effectiveness and superiority.

3. Methodology (of the Students' Project)

This session describes our approach to implement Deep Cluster, the detailed structure, and various technical difficulties our team has encountered during training and implementation.

3.1. Objectives and Technical Challenges

The primary goal of this project is to reproduce the core framework of DeepCluster while adapting it to smaller-scale datasets, such as CIFAR-10, instead of large-scale datasets like ImageNet or YFCC100M, which were utilized in the original paper. Given the computational constraints, such as a single NVIDIA T4 GPU with 16GB of memory [15], we aim to validate the effectiveness of the DeepCluster framework under resource-limited conditions.

Overall, the overall task involves implementing a pseudo-supervised learning approach for image classification using the CIFAR-10 dataset. The specific objectives of this project include:

1) *Dataset Adaptation*: The original Deep -Cluster implementation was evaluated on massive datasets like ImageNet (1.28 million images) and YFCC100M (100 million images). We instead opt for CIFAR-10, a much smaller dataset with 50,000 images. This adjustment enables training and clustering under resource constraints.

2) *Network Simplification*: Replaced complex architectures like AlexNet and VGG16 from the original work with MobileNet and a custom CNN to enhance feature extraction efficiency.

3) *Framework Migration*: Implement the core DeepCluster pipeline using TensorFlow /Keras, while the original implementation relied on PyTorch.

Several technical challenges arise in the process of reproducing and adapting the DeepCluster method:

1) *Low Semantic Complexity*: CIFAR-10, being a small-scale dataset with relatively low -resolution images (32x32 pixels) compared to ImageNet (228x228 pixels), creates challenges in extracting significant features for clustering tasks.

2) *Computational Limitations*: Limited GPU and memory resources restrict the use of large-scale datasets and deep network architectures, necessitating optimized training strategies.

3) *Alternating Training Stability*: DeepCluster relies on alternating between clustering and CNN training. Ensuring convergence and stability in simplified settings with reduced data and model capacity remains a significant challenge.

3.2. Problem Formulation and Design Description

In this section, we present the detailed formulation of the problem and the corresponding system design that addresses the objectives and challenges outlined previously. The system integrates data preprocessing, feature extraction, unsupervised clustering, and model training to achieve this goal. The overall workflow of the project is illustrated in Figure 2. This model consists of the following main components.

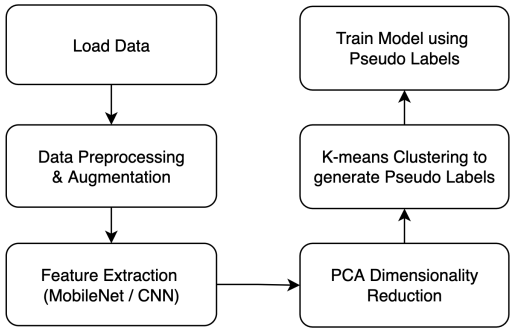

Fig. 2: Overall Workflow of the Project

- **Data Preprocessing**

Unsupervised methods often do not work effectively on color images, as the additional color channels introduce unnecessary complexity without contributing significantly to the task at hand [16]. To address this issue, the original paper used grayscale conversion which is a common strategy in unsupervised learning, as it reduces the input dimensionality while preserving essential structural information such as edges and contrasts [17].

Several strategies exist to remove color and enhance local contrast, such as using fixed linear

transformations based on Sobel filters [18], as described in previous works. Sobel filters emphasize gradients and local features, which are particularly useful for edge detection and improving contrast.

However, in our implementation, we simplify this process by using TensorFlow's built-in grayscale conversion function:

$$x_{\text{gray}} = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$

This linear transformation removes the color information while maintaining a clean representation of image structure.

The grayscale images are then normalized to the range $[0, 1]$ to standardize the input data for the subsequent stages of the pipeline:

$$x_{\text{norm}} = \frac{x_{\text{gray}}}{255}.$$

- **Data Augmentation**

Data augmentation plays a critical role in improving the generalization capability of convolutional neural networks, especially in unsupervised and pseudo -supervised settings [19]. It artificially increases the size and diversity of the training data by applying random transformations to input images. In this work, we employ a randomized augmentation strategy to introduce variability in the training data while preserving the semantic meaning of the images.

Given an input image *x*, the augmentation function randomly applies one of the following trans -formations: 1) No Augmentation, 2) Random Cropping, 3) Horizontal Flipping and 4) Random Brightness Adjustment.

By applying this augmentation function, the model learns features that are invariant to minor spatial and illumination changes, thereby enhancing its generalization performance in the absence of strong supervision.

- **Convnet Feature Extraction**

Given a set of input images $X = \{x_1, x_2, \ldots, x_N\}$, a convolutional network f_θ with parameters θ can be applied to map the input x_n to a lower-dimensional vector representation (features) in a fixed space:

$$z_n = f_\theta(x_n)$$

where $z_n \in \mathbb{R}^d$ represents the feature vector extracted for the input x_n. In our implementation, we employ two types of networks for feature extraction:

MobileNet: A pre-trained, lightweight convolutional neural network that has been fine-tuned for feature extraction. MobileNet is initialized with ImageNet weights and modified to produce a feature vector by applying a Global Average Pooling (GAP) layer and a dense layer.

Custom CNN: A manually designed convolutional network consisting of two convolutional layers with ReLU activations and max-pooling layers, followed by a dense layer to output the features.

This dual-network strategy allows us to compare the performance of a pre-trained model (MobileNet) against a simpler, lightweight architecture (Custom CNN).

- **PCA for Dimensionality Reduction**

Following the feature extraction step using the convolutional neural network f_θ, the resulting features X_{features} often reside in a high-dimensional space. While these high-dimensional features retain valuable information, they are computationally expensive to process and may contain redundant or irrelevant information.

To address this, we follow the original paper to apply Principal Component Analysis (PCA), a widely used linear dimensionality reduction technique. PCA projects the extracted feature vectors $X_{\text{features}} \in \mathbb{R}^d$ onto a lower-dimensional subspace while preserving as much variance in the data as possible. Formally, PCA reduces the dimension d of the feature space to d' components (where $d' < d$).

In our implementation, we set d' = 50 , reducing the dimensionality while retaining most of the variance. This step significantly reduces computational overhead for subsequent clustering and classification tasks.

- **Unsupervised Learning by Clustering**

In supervised learning, the network parameters θ and a classifier g_W with parameters W are optimized jointly to predict the class labels $y_n \in \{0, 1\}^k$, where k is the number of predefined classes. The

optimization goal is to minimize the multinomial logistic loss (negative log-softmax) for a training set of size N:

$$\min_{\theta,W} \frac{1}{N} \sum_{n=1}^{N} \ell\left(g_W\left(f_\theta(x_n)\right), y_n\right), \quad (1)$$

where ℓ is the cross-entropy loss function [5].

However, in the absence of supervision, such parameters cannot be learned directly. Therefore, we turn to unsupervised learning methods to extract meaningful representations.

When the network f_θ is initialized with random parameters θ (e.g., sampled from a Gaussian distribution), the extracted features $f_\theta(x_n)$ do not initially provide discriminative power. Despite this, random convnets still exhibit a degree of structure tied to their convolutional filters, which can act as a weak prior for the input signal.

The goal of our work is to bootstrap this weak prior using unsupervised clustering to iteratively learn both the features and their groupings. Specifically, the extracted features $z_n = f_\theta(x_n)$ are clustered into k groups using the k-means algorithm.

The problem is formalized as learning a centroid matrix $C \in \mathbb{R}^{d\times k}$ and cluster assignments y_n for each image x_n by solving the following optimization problem [5]:

$$\min_{C\in\mathbb{R}^{d\times k}} \frac{1}{N} \sum_{n=1}^{N} \min_{y_n\in\{0,1\}^k} \|f_\theta(x_n) - Cy_n\|_2^2 \quad \text{subject to} \quad y_n^\top \mathbf{1}_k = 1. \quad (2)$$

where:

- $f_\theta(x_n) \in \mathbb{R}^d$ is the feature representation of image x_n,

- $C \in \mathbb{R}^{d\times k}$ represents the k-centroids in the feature space,

- $y_n \in \{0,1\}^k$ is a one-hot assignment vector called pseudo-label that assigns x_n to one of the k clusters,

- $\mathbf{1}_k$ is a vector of ones with dimension *k*.

- **Optimization**

These pseudo-labels (y_n^*) are then treated as supervisory signals to update the parameters θ of the convolutional network. Specifically:

- The network parameters θ are updated by minimizing the classification loss with pseudo-labels y_n^*,
- The updated features $f_\theta(x_n)$ are re-clustered to refine the pseudo-labels.

This alternating procedure iterates between:

- Clustering the feature representations to generate pseudo-labels,
- Training the convolutional network to improve feature quality using the pseudo -labels.

The iterative optimization alternates between solving Equation (2) and updating parameters in Equation (1). This procedure avoids reliance on direct supervision and gradually enhances the discriminative power of the learned features.

In summary, the problem formulation consists of:

Input:

- Input images $X = \{x_1, x_2, \ldots, x_N\}$
- Convolutional network f_θ with initial random parameters θ
- Number of clusters k
- Total epochs E
- Iterations per epoch I_T

Step 1: Data Preprocessing
For each image x_n **in** X **do**:
- Convert image to grayscale.
- Normalize pixel values to [0, 1].
- Apply random data augmentation (crop, flip, brightness).

End For
Step 2: Iterative Learning and Clustering
For epoch = 1 **to** E **do**:
  For iter = 1 **to** I_T **do**:
    Generate a random batch B = x_b from pre -processed train data.
    # 1) Feature Extraction
    For each image x_b **in** batch B:
      Extract features $z_b = f_\theta(x_b)$.
    End For
    # 2) Dimensionality Reduction

Project features using PCA: $z_{\mathrm{PCA}_b} = \mathrm{PCA}(z_b)$

3) Clustering

Perform K-means clustering on z_{PCA_b} to generate pseudo-labels y_b.

4) Forward Pass (FP) and Backward Pass (BP)

Treat pseudo-labels y_b as supervision:

- Compute loss:
  $L = \mathrm{CrossEntropyLoss}(f_\theta(x_b), y_b)$
- Update network parameters θ using Adam optimizer:
  $\theta \leftarrow \theta - \eta \cdot \nabla_\theta L$
  with $\eta = 0.0001$ and decay $= 1 \times 10$

End For

End For

Output:

- Refined network parameters θ*
- Final pseudo-labels y_n for all images

4. Implementation

In this section, we will first introduce the dataset, then explain the architecture of our network and show the algorithm network design detail.

4.1 Data

This project used the CIFAR-10 dataset for both training and testing.

The CIFAR-10 dataset is a small-scale dataset used for image classification tasks. It consists of 60,000 color images across 10 categories, with 50,000 images for training and 10,000 images for testing. The images have a low resolution of 32×32 pixels. The dataset covers multiple types of objects, including airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

After loading the CIFAR-10 dataset, it is divided into training and testing sets, with input images and labels separated. To simplify the model input, images are converted to grayscale to reduce input dimensionality, and the data format is transformed into Tensors. To enhance data diversity, a set of random data augmentation strategies is applied, including random cropping (cropping the image to 28×28 and resizing it back to 32×32), random horizontal flipping, random brightness adjustment (slight modifications to brightness), and retaining the original image without changes. Finally, the image data is normalized.

4.2 Deep Learning Network

1) Architectural Block Diagrams:

Figure 3 illustrates the overall architecture of our implementation. The process begins with the Feature Extraction module, where input data is passed through a deep convolutional neural network to extract high-dimensional features. These features are then reduced using PCA to simplify the representation while retaining key information. The reduced features are clustered using K-means, generating pseudo-labels that serve as supervisory signals for training the classification module.

The Classification module utilizes the pseudo-labels to perform a supervised learning task. The classification loss is propagated backward through the network, updating both the classification head (top layer) and the feature extraction module. This iterative process enhances the feature representations, creating a feedback loop between feature extraction and clustering.

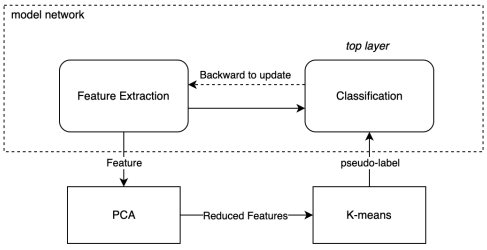

Fig. 3: An illustration of the architecture of DeepCluster

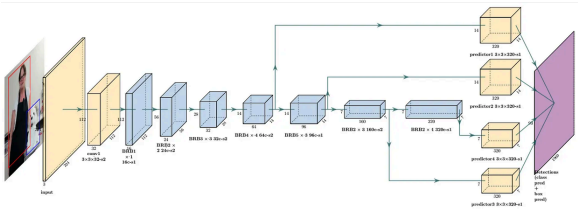

Fig. 4: An illustration of the architecture of MobileNet

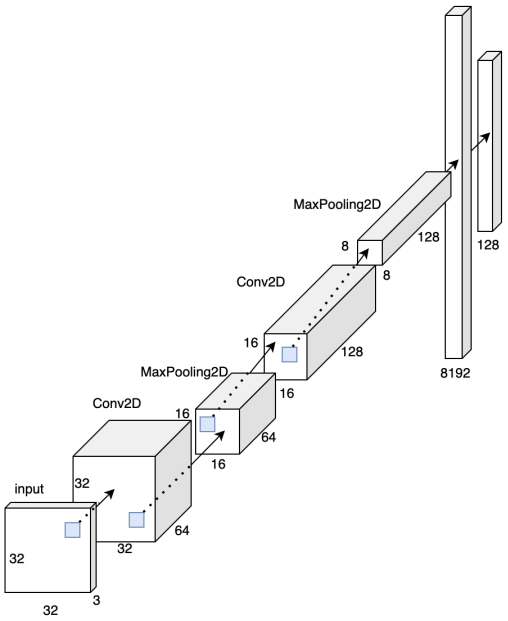

Fig. 5: An illustration of the architecture of Custom CNN

i) Feature Extraction: The implemented architectures for feature extraction include MobileNet and a Custom Convolutional Neural Net- work (CNN).

The MobileNet model, illustrated in Figure 4 [20], utilizes a pre-trained MobileNet architecture with ImageNet weights. The top classification layer is removed and replaced with a Global Average Pooling (GAP) layer and a Dense layer with 512 units and L2 regulari- zation ($\lambda = 0.001$) to mitigate overfitting. Layers 1–20 are frozen to retain learned features, while deeper layers remain trainable for fine-tuning.

The Custom CNN architecture, as shown in Figure 5, consists of two convolutional layers followed by MaxPooling operations, a Flatten layer, and a fully connected Dense layer. It processes input of size $32 \times 32 \times 1$, using 3×3 convolution kernels with ReLU activation and 'same' padding to preserve spatial dimensions. MaxPooling reduces feature map size, and the final Flatten layer transforms the output into a vector of size 8192, followed by a Dense layer with 128 units.

ii) Classification: The classification module is designed to map the extracted features to the target pseudo-labels obtained through clustering. Specifically, a Dense layer with a softmax activation function is added as the top layer of the model, where the number of output neurons corresponds to the number of clusters (n_clusters). The softmax activation ensures that the output represents a probability distribution across the cluster categories.

This module uses the output from the feature extractor as input and produces class probabilities, which are later used to compute the classification loss. Through backpropagation, the classification loss optimizes both the classification head and the preceding feature extraction network, refining the overall feature representations iteratively.

2) Training Algorithm Details:

The training process for the deep learning network involves compiling the model with the Adam optimizer, a learning rate of 0.0001, and weight decay of $1e-4$. The Sparse Categorical Cross-entropy loss function is used alongside accuracy as the evaluation metric. The dataset is split into training (80%) and validation (20%) sets, with data augmentation applied to the training set, and both datasets are batched using `tf.data` pipelines with a batch size of 256. The network is trained for 40 epochs using pseudo-labels generated from the clustering process.

To enhance convergence and prevent overfitting, ReduceLROnPlateau dynamically reduces the learning rate by a factor of 0.5 if the validation loss does not improve for 3 consecutive epochs, and Early Stopping is implemented with a patience of 5 epochs, restoring the best model weights based on the validation loss. During training, the model iteratively optimizes both the feature extraction layers and the classification head using backpropagation, refining the feature representations guided by clustering results.

4.3 Software Design

1) Data Preprocessing & Augmentation

The input data consists of images with diverse classes and varying resolutions. Initially, the input images were converted to grayscale to reduce computational complexity while retaining essential information. The images were normalized and resized to 32x32 pixels, which aligns with the requirements of the neural network input layer [21]. Then, a data augmentation strategy was applied to increase the dataset variability and prevent overfitting. The augmentation process randomly selected one of the following trans- formations: no augmentation, random cropping to 28x28 pixels followed by resizing back to 32x32,

horizontal flipping, or brightness adjustment. This randomized augmentation effectively introduces diversity into the training set while preserving the core image features.

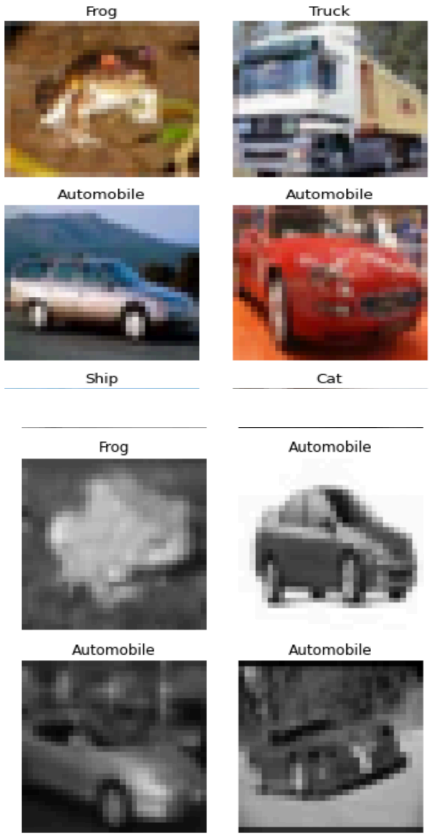

Fig. 6: Examples of Original Images (top) and Preprocessed & Augmented Images (bottom)

2) Pseudo-Supervised Classification

Model Selection: To facilitate effective feature extraction, we implemented two model options for comparison and selection: MobileNet and a Custom CNN.

PCA: To reduce the dimensionality of extracted features, PCA is applied to the input feature set, compressing the data to 50 dimensions while preserving most of the variance. This step ensures computational efficiency and removes redundant information from the feature space.

K-means: The K-means clustering algorithm is employed to partition the reduced features into n-clusters, where n corresponds to the predefined number of classes (e.g., 10 for CIFAR-10). The clustering process iteratively assigns data points to the nearest cluster centroids, minimizing intra-cluster variance. The resulting cluster assignments are used as pseudo-labels for subsequent supervised training. The output pseudo-labels serve as a bridge between feature extraction and classification, enabling the model to learn meaningful representations in a self-supervised manner. The entire process ensures that the generated pseudo-labels approximate the underlying class structure of the dataset.

Backpropagation: Following the training algorithm previously detailed in Section 4.2, the pseudo-labels generated by K-means clustering are used as supervision to train the classification network. These pseudo-labels are fed into the classifier, which computes the cross-entropy loss. The loss is then backpropagated through the network, updating the weights of the feature extractor and the classifier.

Iteration: The updated network is subsequently used to re-extract features, iterating through the same process of PCA and clustering to produce refined pseudo-labels. This iterative framework enables the model to progressively learn more discriminative features and improve the clustering quality. By combining feature extraction, clustering, and classification into a unified pseudo-supervised learning loop, the network aligns its learned representations with the underlying structure of the dataset, even in the absence of ground truth labels.

Model Save & Load: The trained models are saved and loaded for further evaluation and deployment. The MobileNet and Custom CNN models are stored in the 'model' folder. They can be reloaded using the *load_model* function, enabling seamless continuation of training, validation, or inference.

5. Results

5.1 Visualizations

5.1.1. First Layer filters

The filter weights learned by the first layer are randomly showing gray or black mode, no obvious edge features are detected. The second figure presents the activation outputs of the first layer filters (conv1) for a Mobilenet trained with DeepCluster with grayscale transformation images. The activation results demonstrate that the first layer filters primarily capture smooth regions, low-frequency textures, and edge detection features.

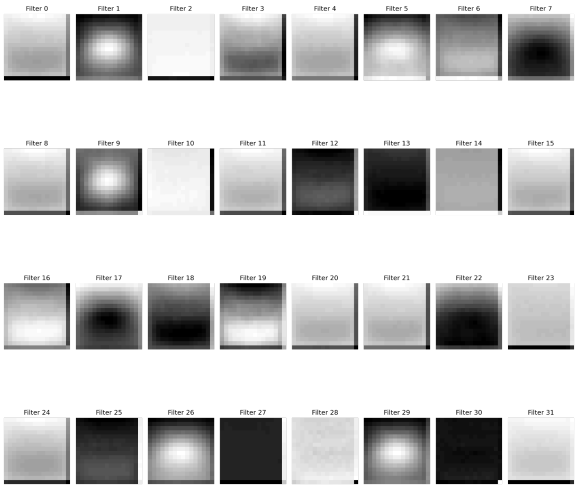

Fig.7: Activation results of first convolutional layer

Some filters exhibit Gaussian-like patterns with a high-intensity response in the center (e.g., filters 9, 26, and 29), suggesting they respond strongly to local brightness variations and capture a strong local brightness peak with smooth gradients in all directions. Other filters (e.g., filters 5, 12, 22) respond to edges or high-contrast regions in specific orientations, resembling basic edge detection operators. Certain filters (e.g., filters 3, 4, 11, 19) display relatively uniform activations across the entire image, indicating limited feature extraction capability.

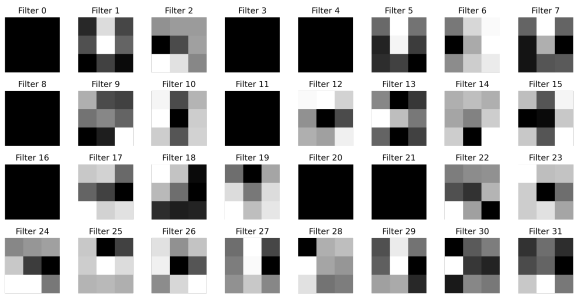

Fig.8: Visualization of first convolutional layer weights.

Comparison: The original work compared filters from the first layer of an AlexNet trained with DeepCluster on raw RGB images and images preprocessed with Sobel filtering, concluding that color plays little role in object classification. Therefore, we trained our model using grayscale-converted data. Due to device limitations, we trained the model on a smaller dataset and in a lighter manner. Our analysis primarily focuses on the features captured by the activated filters in the first convolutional layer because the results of direct weights visualization were relatively poor (See Figure 8).

5.1.2. Probing deeper layers

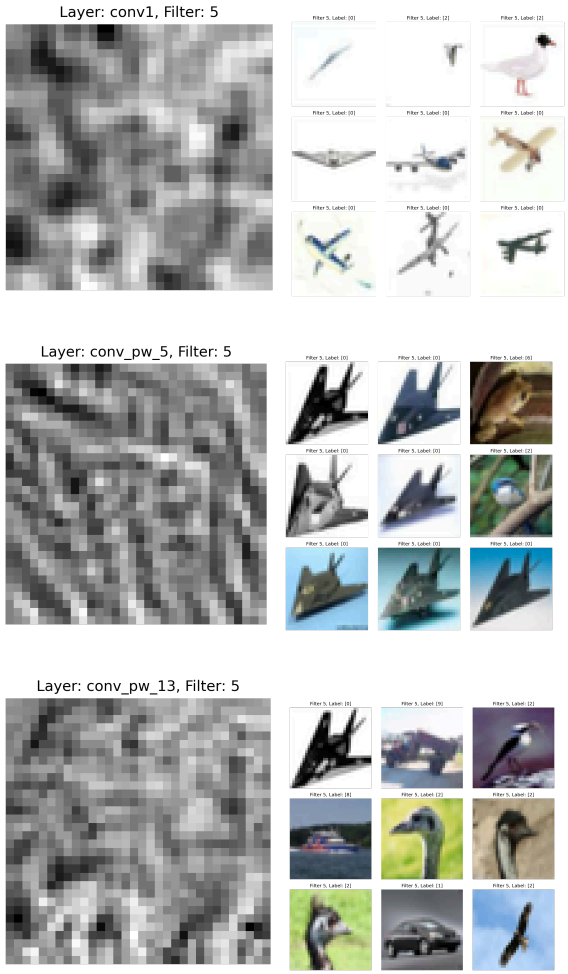

Fig.9: Filter visualization and top 9 activated images from a subset of 50000 images from CIFAR10 for target filters in the layers conv1, conv5 and conv 13

We assessed the quality of a target filter by learning an input image that maximizes its activation. We employed two visualization methods to assess the quality of learned filters in our network. First, we utilized gradient ascent to generate synthetic images that maximize the activation of specific filters, revealing the patterns or features to which the filters are most responsive. Second, we activated the network with real dataset images to observe the filter responses and identified the top 9 images that elicited the strongest activations, providing insight into the

types of structures or objects captured by the filters in our dataset.

Following the same methods adopted in the original paper, we used the gradient ascend visualization method proposed by Yosinki *et al*. [22] We optimize a random input image to maximize the activation of a specific filter in a neural network layer, iteratively updating the input using gradients and regularization to generate interpretable visualizations of the learned features. This method reveals the patterns or features to which the filter is most responsive to.

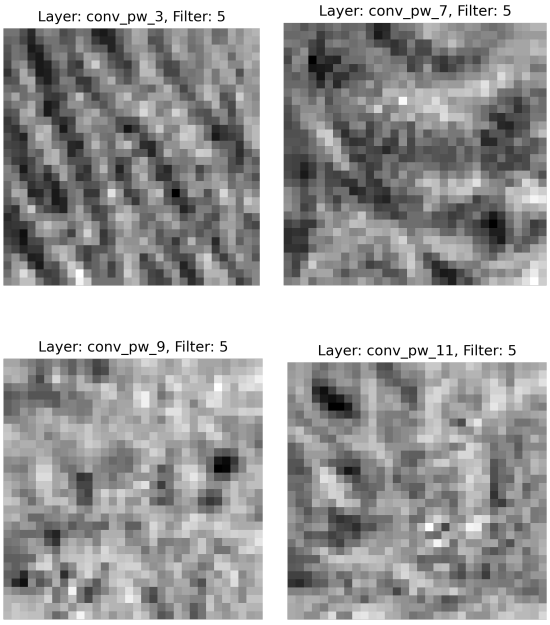

Fig.10: Visualization of Activation Patterns for Intermediate Convolutional Layers

The visualization of a target filter showed coarse and pixelated patterns. Our model's filters successfully learned some low-level edges, textures, and generic patterns, but they still lacked capturing clearly defined higher-level structures and interpretable structures.

We found that in the shallow layers of the network, filters learned coarse and blurry features, which indicates the filters are struggling to specialize in edge or contrast detection features. The filter learned more textured and complex patterns in deeper layers (conv3 and conv5). However, the refining was consistent as the layer went deeper; in layers conv7 to conv13, little improvement was shown, and the features even showed less interpretability. Some layers tend to replicate previously captured textures.

Comparison: Our model's pattern was similar to the original paper to some extent; deeper layers (conv3, conv5) in our network seem to capture larger textural structures than shallow layers (conv1). The last several layers only had minor improvements. The original paper's networks showed highly structured and intricate patterns. The filters in the shallow layers of their network showed well-defined edge detection and clear textual patterns, and deeper layers captured larger textural structures and geometric shapes. However, the patterns captured by our model's deeper layers (e.g., conv3 and beyond) lack the richness and complexity seen in the original paper's visualizations because of the smaller dataset and network size, leading to limited learning capacity in our experiment.

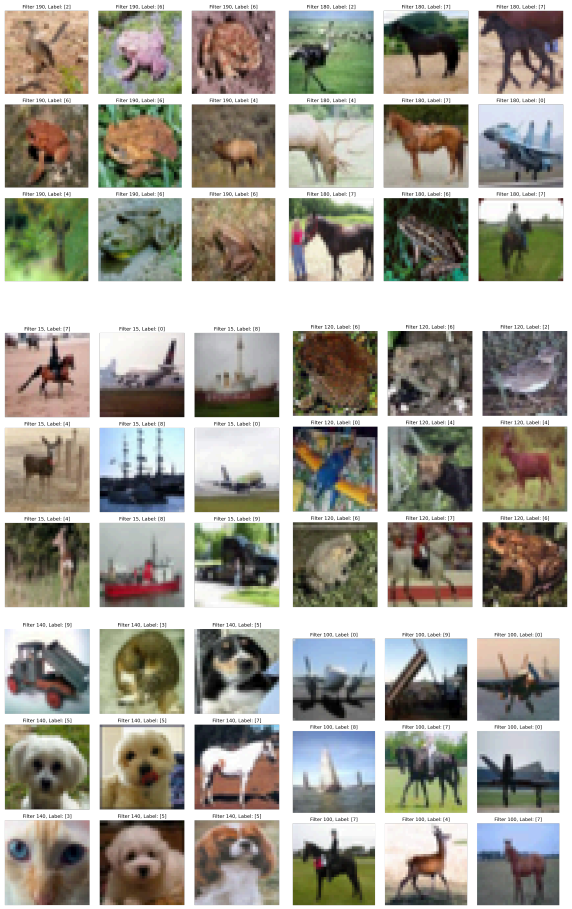

Fig.11: Top 9 activated images from the CIFAR-10 dataset for selected filters in the final convolutional layers of our MobileNet-based model. The results show filters that activate strongly for images containing specific objects or shapes, such as animals or vehicles. For instance, filter 100 is sensitive to structural patterns in airplanes and landscapes.

Finally, Figure 11 shows the top 9 activated images for some conv13 filters that appear to be semantically coherent. Most filters have learned to capture structures that strongly correlate with object classes, as evidenced by a significant tendency for a single filter to activate in response to specific object types at the shape level. However, distinguishing finer details remains challenging. For instance, filter 140 predominantly activates for animal heads, but it also incorrectly includes a trunk in its Top 9 activations.

Our results indicate that the first convolutional layer of the model trained on grayscale data successfully learns basic edge, brightness, and low-level texture features.

5.2 Model Evaluations

We trained a MobileNet model on the CIFAR-10 dataset using our proposed method, which iteratively applies k-means clustering to the features extracted by the convolutional network and updates the model's weights by treating the cluster assignments as pseudo-labels in a discriminative loss framework.

We set 40 epochs with early stopping with the best training accuracy of 81.00%, the best validation accuracy reaching 80.27%, minimum training loss of 0.5134, and minimum validation loss of 0.5328 after early stopping. For evaluation, the grayscale input data was converted into RGB format to match MobileNet's input requirements. We used the model to make predictions and tested it against the ground truth labels; the model achieved an accuracy of 12.09%**.** This result indicates a limited capacity of the model to generalize features for true classification tasks.

5.3 Linear Classification on Activations

Identically, we performed a linear classification task as the original paper's work. This analysis aims to identify at which layer the convnet becomes specialized in object classification. According to Zhang *et al*. [23], we train a linear classifier on top of different frozen convolutional layers. We extracted activations from the first five convolutional layers of the network and trained it on a subset of CIFAR-10 dataset due to the limitation of computational resources. A logistic regression classifier was trained on the extracted activations from the training set and evaluated on the test set. We applied Principal Component Analysis (PCA) to reduce the dimensionality of the activations, with the number of components set to 30. We recorded the accuracy of the linear classifier trained on the activation by layer, and reported the results of our experiment and compared it to the original paper model's performance.

On CIFAR-10, DeepCluster achieved 1.19% ~ 22.12% better performance than the state of the art for conv1 - conv5] layers. We observed the most significant improvement at the conv1 - conv2 layer and conv3 - conv4, reaching 22.12% and 19.6% differences respectively. While the difference between conv2 - conv3 only be 5.7%. Similar to the original paper's pattern, the conv1 layer underperformed but then had a sharp enhancement in conv3 - con5. A notable improvement is observed in the deeper layers, with conv3 achieving 49.30% and conv5 peaking at 59.70%, indicating that deeper layers encode more class-specific and semantically meaningful information. Interestingly, the conv2 and conv3 layers show similar accuracy of 46.65% and 49.30%. Therefore, the results suggested that mid-level representations (conv3-conv5) contribute most to classification tasks.

Our findings are consistent with the results reported by the original work that the AlexNet and MobileNet probably store most of the class level information at conv3 - conv5. Our layer-wise accuracy is higher than original papers' probably because of the smaller and simpler dataset.

5.4 Discussions

The visualization results indicate that the trends between our MobileNet-based model and the original paper's AlexNet are somewhat aligned: deeper layers capture more features and details compared to shallower layers. Our shallow layers primarily captured basic edges, brightness variations, and low-frequency textures, while the deeper layers showed more complex and textured patterns. However, the features learned by our model are less clear and less diverse than those observed in the original work. This discrepancy is likely due to the differences in the dataset and model architecture. The original paper trained AlexNet on significantly larger and more diverse datasets (e.g., ImageNet or YFCC100M), whereas our model was trained on the much smaller CIFAR-10 dataset, which contains only 50,000 images with limited variability. Additionally, AlexNet's higher capacity (e.g. five convolutional layers using 96 to 384 filters per layer and three fully

connected layers) and depth may contribute to its ability to capture richer and more semantically meaningful features.

In the classification task, the layer-wise results showed that deeper layers (conv3-conv5) contributed the most to class-specific features, with the highest accuracy achieved at the conv5 layer (63.05%). This pattern aligns with the findings in the original paper, where mid-level layers are identified as critical for encoding semantically meaningful features. Additionally, the accuracy values achieved by our model were higher than those reported in the original paper. This counterintuitive result can also be explained by the differences in datasets: CIFAR-10 is simpler and less diverse than ImageNet, which reduces the complexity of the classification task and allows even less sophisticated features to perform well.Additionally, AlexNet's higher capacity (i.e. five convolutional layers using 96 to 284 filters per layer and three fully connected layers) provides a significantly larger parameter space compared to our MobileNet-based model, which uses depthwise separable convolutions to reduce computational cost.

Our results highlighted the critical role of dataset scale and diversity in determining the generalization capacity of a model. The smaller CIFAR-10 dataset restricted the diversity of features that could be learned, while the lightweight MobileNet architecture, optimized for small-scale tasks, may have hindered the model's ability to capture complex and highly structured features. This is evident in the weaker performance of deeper layers in both visualization and classification tasks compared to the AlexNet results in the original paper. Nonetheless, the simpler dataset also allowed our model to achieve higher classification accuracy, showcasing how task and dataset alignment can influence performance outcomes.

6. Future Work

Our experiments provided two insights for future directions. One key idea is improving the performance of clustering during pseudo-label assignment. Notably, the accuracy of pseudo-labels compared to true labels remains low in both the original projects and our experiments. For instance, the NMI between cluster assignments and ImageNet labels during training is below 50%, highlighting the need for methods to improve generalizability.

Another important direction is addressing the reliance on network structure and dataset size. While self-supervised learning has demonstrated strong performance on large datasets, adaptations for smaller datasets are worth exploring to enhance results. This could involve designing network structures that better capture the unique characteristics of small-scale datasets or applying additional regularization techniques to mitigate overfitting. Furthermore, hybrid approaches that combine self-supervised learning with minimal supervision may offer a promising solution to narrow the performance gap between small and large datasets.

7. Conclusion

In this project, we replicated the experiment of the DeepCluster algorithm with a small-scale dataset and a more efficient network structure. The original work introduced a model tested on large-scale datasets (e.g., ImageNet) and demonstrated significant improvements in unsupervised learning performance. In our work, we trained a MobileNet on the smaller-scale CIFAR-10 dataset using pseudo-labels assigned by a k-means clustering algorithm on extracted features, updating the model's weights with a discriminative loss. Our works assessed the feasibility and effectiveness of applying DeepCluster's unsupervised learning framework in a resource-constrained environment, contrasting its outcomes with those reported for AlexNet on larger datasets.

We found that, while filters in the networks successfully detected some general features and edges, the visualization results lacked interpretability. The model converged very early, and the quality of the extracted features remained limited in terms of clarity and semantics. Our experiments revealed that DeepCluster's performance is highly sensitive to both the network structure and the dataset. The visualization results and model accuracy highlight the inherent limitations of unsupervised clustering methods when constrained by data scale and model capacity.

8. Acknowledgement

We would like to thank Professor Zoran Kostic for his guidance and support throughout the semester and this project. We are also sincerely grateful to the teaching assistants for their invaluable assistance. Additionally, we appreciate the resources provided by

Google Cloud Platform, which were instrumental in our training process.

9. References

[1] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NIPS. (2015)

[2] Kovashka, A., Russakovsky, O., Fei-Fei, L., Grauman, K., et al.: Crowdsourcing in computer vision. Foundations and Trends in Computer Graphics and Vision 10(3) (2016) 177–243

[3] Misra, I., Zitnick, C.L., Mitchell, M., Girshick, R.: Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels. In: CVPR. (2016)

[4] Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations
and image clusters. In: CVPR. (2016)

[5] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep Clustering for Unsupervised Learning of Visual Features," Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 132-149.

[6] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. (2009)

[7] Thomee, B., Shamma, D.A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., Li, L.J.: The new data and new challenges in multimedia research. arXiv preprint arXiv:1503.01817 (2015)

[8] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012)

[9] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale
image recognition. arXiv preprint arXiv:1409.1556 (2014)

[10] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. Accessed: Dec. 18, 2024.

[11] A. G. Howard, M. Zhu, B. Chen, et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv, 2017. [Online]. Available: https://arxiv.org/abs/1704.04861. Accessed: Dec. 18, 2024.

[12] M. Abadi, P. Barham, J. Chen, et al., "TensorFlow: A system for large-scale machine learning," in Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16), Savannah, GA, USA, Nov. 2016, pp. 265–283. [Online]. Available: https://www.tensorflow.org/. Accessed: Dec. 18, 2024.

[13] X. Wang and A. Gupta, "Unsupervised Learning of Visual Representations Using Videos," Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2794-2802.

[14] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised Visual Representation Learning by Context Prediction," Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1422-1430.

[15] NVIDIA Corporation, "NVIDIA T4 Tensor Core GPU," [Online]. Available: https://www.nvidia.com/en-us/data-center/tesla-t4/. Accessed: Dec. 18, 2024.

[16] Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: ICCV. (2015)

[17] A. Dosovitskiy, P. Fischer, J. T. Springenberg, et al., "Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 9, pp. 1734–1747, Sep. 2016. [Online]. Available: https://arxiv.org/abs/1406.6909. Accessed: Dec. 18, 2024.

[18] Bojanowski, P., Joulin, A.: Unsupervised learning by predicting noise. ICML (2017)

[19] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," arXiv preprint arXiv:1712.04621, 2017. [Online]. Available: https://arxiv.org/abs/1712.04621. Accessed: Dec. 18, 2024.

[20] The Modern Scientist, "MobileNet: Revolutionizing mobile and edge computing with efficient neural networks," Medium, Jun. 15, 2023. [Online]. Available: https://medium.com/the-modern-scientist/mobilenet-revolutionizing-mobile-and-edge-computing-with-efficient-neural-networks-5a2cd01a4e47. Accessed: Dec. 18, 2024.

[21] SZQA Team, "E4040-2024fall-project-SZQA-zs2699-yq2411-jz3849," GitHub repository. [Online]. Available: https://github.com/ecbme4040/e4040-2024fall-project-SZQA-zs2699-yq2411-jz3849/tree/main. Accessed: Dec. 18, 2024.

[22] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579 (2015)

[23] Zhang, R., Isola, P., Efros, A.A.: Split-brain autoencoders: Unsupervised learning by cross-channel prediction. arXiv preprint arXiv:1611.09842 (2016)

9. Appendix

9.1 Individual Student Contributions in Fractions

	zs2699	yq2411	jz3849
Last Name	Shi	Qiu	Zhu
Fraction of (useful) total contribution	1/3	1/3	1/3
What I did 1	Implemented the model	Visualized the model weights and the activated results	Implemented the code to download, extract the dataset
What I did 2	Implemented the training algorithm	Performed linear classification tasks to evaluate the model performance	Data preprocess
What I did 3	Evaluate the model via accuracy		Data augmentation