# Synthesized Feature based Few-Shot Class-Incremental Learning on a Mixture of Subspaces

Ali Cheraghian[*,1,2], Shafin Rahman[*,3], Sameera Ramasinghe[1,2] , Pengfei Fang[1,2], Christian Simon[1,2]
Lars Petersson[1,2], Mehrtash Harandi[2,4]
[1]Australian National University, [2]Data61-CSIRO, Australia
[3] North South University, Dhaka, Bangladesh, [4]Monash University, Australia
{Ali.Cheraghian, Sameera.Ramasinghe, Pengfei.Fang, Christian.Simon}@anu.edu.au,
shafin.rahman@northsouth.edu, Lars.Petersson@data61.csiro.au, mehrtash.harandi@monash.edu

## Abstract

*Few-shot class incremental learning (FSCIL) aims to incrementally add sets of novel classes to a well-trained base model in multiple training sessions with the restriction that only a few novel instances are available per class. While learning novel classes, FSCIL methods gradually forget base (old) class training and overfit to a few novel class samples. Existing approaches have addressed this problem by computing the class prototypes from the visual or semantic word vector domain. In this paper, we propose addressing this problem using a mixture of subspaces. Subspaces define the cluster structure of the visual domain and help to describe the visual and semantic domain considering the overall distribution of the data. Additionally, we propose to employ a variational autoencoder (VAE) to generate synthesized visual samples for augmenting pseudo-feature while learning novel classes incrementally. The combined effect of the mixture of subspaces and synthesized features reduces the forgetting and overfitting problem of FSCIL. Extensive experiments on three image classification datasets show that our proposed method achieves competitive results compared to state-of-the-art methods.*

## 1. Introduction

In many practical applications, it is crucial for a model to classify novel objects, *i.e*., objects for which only a few instances are available during training. For example, this can occur when the distribution of the test data deviates from that experienced at training, or if the model faces new objects from a class for which significantly less data was provided during training. While the former, to some extent, can be addressed by various techniques such as domain adaptation, addressing the latter is usually studied under the Few-Shot Learning (FSL) paradigm [33, 10]. Generally, in a FSL framework, the goal is to classify samples into few-shot classes given only a training set of base categories. However, some variations of the problem classify both base and novel class instances together in a generalized manner. In a more realistic scenario, all novel class instances may not be available at a time. It creates another branch of the problem, few-shot class incremental learning (FSCIL), where novel classes are added to the model incrementally over time, and in each incremental step, the model is tested based on both base and novel class instances. Because of this restriction, FSCIL is the most complex form of FSL problem.

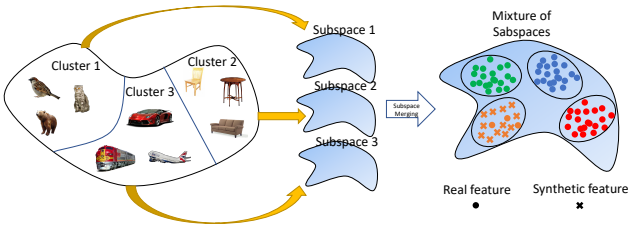



Figure 1: We cluster the training samples into a number of clusters, in this example, three. Then, we generate three corresponding subspaces, which are constructed from these three clusters. These subspaces are then used to project visual features and semantic vectors onto. The distance between the projected visual feature and the projected semantic vector is minimized according to a loss function during training in each subspace. In addition to projecting real sample features, we also project synthesized features to the mixture of subspaces. The combined effect of the mixture of subspaces and synthesized features helps the network not overfit to few-shot data of novel classes and forget base class knowledge.

Initial results on FSCIL have been proposed in the literature [35, 20, 4, 8]. We identify two critical challenges in this problem. ***(a)*** Catastrophic forgetting of base classes: Recent works observed a fascinating performance using word

[*]denotes equal contribution.

vectors in the learning process (in addition to knowledge distillation techniques) to address the forgetting problem [25, 8, 49]. The general motivation is that shared attributes (e.g., shape, color) between the base (e.g., horse, tiger) and novel class (e.g., zebra) help to understand novel classes better using a few examples and not to forget base classes. Considering the subtle variations of such relationships between closely and distantly related classes, [8] proposed to apply clustering on semantic vectors of the embedding space. However, being trained on noisy unsupervised texts, word vectors always estimate only a crude stereotype of any class name, not truly reflecting dataset specific visual interplay between base and novel objects. In this paper, we argue that the relation of visual and semantic vectors must be computed on an embedding space which has the knowledge of the entire dataset. ***(b)*** Overfitting to novel classes: To address the problem, traditional methods use class prototypes [11, 27] and some memory [8] of base class instances. However, because of fewer data available to train novel classes incrementally, it is not easy to escape from this problem. Moreover, if any intermediate incremental step faces this problem, the impact propagates in future incremental trials. In this paper, we argue that including synthesized features of novel classes can reduce this problem.

We endeavor to design an FSCIL approach that improves the classification performance while not suffering from the drawbacks of the methods mentioned in the above paragraph. Here, we also use semantic word vectors in the network pipeline. We apply clustering in image feature space instead of word embedding space while relating similar and dissimilar base classes of a novel class. Based on each cluster, we create a set of subspaces. The subspaces are constructed in such a way as to best represent individual clusters of features formed by visually similar samples. Singular Value Decomposition (SVD) is employed for this purpose, and by selecting a set of basis vectors with the greatest eigenvalues, we ensure that the signal in each visual feature cluster is well represented. Less prominent portions of the feature clusters are more likely noise than signal and will, hence, not be well represented in the subspace. Empirically, we observed that capturing information about how the data projects onto such subspaces leads to less forgetting of base classes and better alignment of features and semantics of classes. Next, at each incremental step, we utilize a variational auto encoder (VAE) for producing high-quality synthesized features representing rich prior knowledge about novel classes. The generative model is trained using only available class instances, capable of generating and augmenting novel class features using a few examples during each incremental session. Note that instead of the traditional use of semantic word vectors in such a feature generation process [15, 29, 39, 43, 14], here we use sampled features to generate more features. Considering the mixture of subspaces while relating base and novel classes and augmenting synthesized features at each incremental session reduces both catastrophic forgetting and overfitting problems during novel class training. Evaluating on MiniImageNet, CUB200, and CIFAR100 cloud benchmark datasets, we consistently outperform many current state-of-the-art methods.

In summary, the contributions of this work are: **(1)** a novel FSCIL framework that elegantly addresses both the catastrophic forgetting problem of base classes by using a mixture of subspaces and the overfitting problem of novel classes by using synthesized features. **(2)** a subspace computation strategy based on clustering in image feature space to relate base and novel classes more accurately for the FSCIL problem. **(3)** state-of-the-art performance on MiniImageNet, CUB200, and CIFAR100 cloud benchmark datasets.

## 2. Related work

**Incremental learning:** Incremental learning methods are divided into three groups, task-incremental learning [3, 28, 24], domain-incremental learning [48, 31], and class-incremental learning [26, 2, 13, 40]. We focus only on the class-incremental learning problem . Rebuffi *et al*. [26] maintains an "episodic memory" of the instances. Additionally, they incrementally accommodate the nearest-neighbor classifier for the new tasks. Castro *et al*. [2] use a knowledge distillation loss to store knowledge about previously seen concepts, and a classification loss is applied to learn the new concepts. Hou *et al*. [13] proposed an innovative approach for incrementally learning a unified classifier that decreases the imbalance between old and new classes by cosine similarity. Wu *et al*. [40] adjust the bias in the model's output with the aid of a linear model. In this paper, similarly, we propose a class-incremental learning method that works on the low data regime.

**Few-shot class incremental learning:** FSCIL was introduced by Tao *et al*. [35] for the first time. They use a neural gas (NG) network to reduce the catastrophic by learning and maintaining the topology of the feature generated by different classes. Mazumder *et al*. [20] choose a few model parameters to learn every novel set of classes rather than training the full model, which helps prevent overfitting. Additionally, by holding the essential parameters in the model intact, they minimize catastrophic forgetting. Chen *et al*. [4] propose a nonparametric approach in deep embedded space. They compress the information of the learned tasks within a tiny amount of quantized reference vectors. They include intra-class variation, less forgetting regularization, and calibration of reference vectors to mitigate catastrophic forgetting. Cheraghian *et al*. [8] utilize word vectors with a distillation method to reduce the effect of catastrophic forgetting. Moreover, they use an attention mechanism to reduce the overfitting issue on novel classes, where there are only

a few training samples available for them during training. [32] generates one subspace per class, where each subspace is the only representative of a particular class. Our method creates multiple subspaces based on the cluster structure of the entire training dataset and remains shared among all classes. [45] uses multiple randomly initialized embeddings. To make these embeddings different, they used unlabeled test data. In contrast, we generate multiple subspaces based on the training data distribution. Each subspace is created based on one part of the training distribution. As a result, each subspace is unique.

**Learning without forgetting using word vectors:** Word vectors have shown promising success on various computer vision tasks such as zero-shot learning, few-shot learning, image/video captioning and visual question answering [15, 29, 39, 43, 9, 46, 47, 6, 5, 7]. Lately, some works [25, 8, 49] have shown word vectors can likewise be beneficial for learning without forgetting. Rahman *et al*. [25] has used semantic word vectors in the any-shot object detection problem in order to detect both unseen and few-shot objects simultaneously. Word vectors helped to reduce the forgetting of seen classes during fine-tuning. Cheraghian *et al*. [8] used word vectors for the FSCIL problem in their proposed pipeline to reduce catastrophic forgetting. They use a distillation method to address the forgetting issue and use of semantic word vectors during the training stage. Zhu *et al*. [49] use word vector for the few-shot object detection problem. They introduce a method that learns new objects from both the visual information and the semantic relation. Notably, they form a semantic space employing the word embeddings, where the detector is trained to project the objects from the visual domain to the semantic domain. This paper uses both visual and semantic class information to form class prototypes on a mixture of subspaces defined on base class instances.

**Generative model for synthesized feature:** Synthesizing features to improve the performance of deep classification networks has been an interesting approach practiced in several recent works. In challenging scenarios where limited or no data is available, generating artificial features helps the models cope with the extreme imbalance in training data. For instance, Xian *et al.* [42] employed a generative adversarial network (GAN) to synthesize features using class-level semantic information. They utilized these features in a zero-shot learning setting and affirmed that the generated features consist of sufficient discriminative properties for training softmax classifiers or any multimodal embedding method. In contrast, Schonfeld *et al.* [30] used a VAE for the same purpose. However, as opposed to [42], they enforce the VAE to learn a shared latent space of image features and class embeddings, making the VAEs sensitive to the modality. Afterward, the learned latent features are used to train a softmax classifier. Similarly, Xian *et al.* [44] tackle the any-shot learning setting, i.e., zero-shot and few-shot, in a unified feature generating framework that operates in both inductive and transductive learning settings. They introduce a conditional generative model that fuses the ability of both VAE and GANs, where the model learns the marginal feature distribution of unlabeled images via an unconditional discriminator. In contrast to aforesaid models, we do not utilize semantic embeddings to generate visual features, and only use the available (limited) visual features to train the generative model.

# 3. Method

## 3.1. Problem Formulation

Given a sequence of tasks $\mathcal{Q} = \{\mathcal{Q}^1, \ldots, \mathcal{Q}^T\}$, where $\mathcal{L}^t$ is the set of classes in the task $\mathcal{Q}^t$, and $\mathcal{L}^i \cap \mathcal{L}^j = \phi$, $\forall i, j \in \{1, \ldots, T\}$, where $i \neq j$. Moreover, a set of $d$-dimensional semantic class embeddings for each class label of all tasks are defined as $\mathcal{E}^t$. We define a task set $\mathcal{Q}^t = \{(\boldsymbol{x}_i^t, l_i^t, \boldsymbol{e}_i^t)\}_{i=1}^{N_t}$, where $\boldsymbol{x}_i^t$ is the $i^{\text{th}}$ sample with the label $l_i^t \in \mathcal{L}^t$, $\boldsymbol{e}_i^t \in \mathcal{E}^t$ is its corresponding semantic class embedding, and $N_t$ is the number of samples. In the FSCIL setting, there are many training instances available for the first task, i.e., the base task $\mathcal{Q}^1$. In contrast, only a few training instances (5-shot per class) are available for the other tasks, i.e., novel tasks $\{\mathcal{Q}^2, \ldots, \mathcal{Q}^T\}$. It is critical to mention that only the training instances of the $t$-th task is observed by the model during training of this task. During inference, the trained model on the current task $\mathcal{Q}^t$ should predict the output for test instances belonging to both $\mathcal{Q}^t$ and all the previous tasks $\{\mathcal{Q}^1, \ldots, \mathcal{Q}^{t-1}\}$.

## 3.2. Model Overview

Our proposed architecture is illustrated in Fig. 2. Class information of the visual and semantic domains is aligned within each subspace such that labels for all the task's instances can be predicted. An image $\boldsymbol{x}_i$ is fed into a CNN (*e.g.*, ResNet-18 [12]), which is trained only on the first task $\mathcal{Q}^1$. The CNN backbone output (before the last layer) is used to extract a visual feature representation $\boldsymbol{y}_i \in \mathbb{R}^m$. For the following tasks $\mathcal{Q}^t$ where $t > 1$, the backbone remains unchanged. Similar to the backbone, the VAE block generates a visual feature $\boldsymbol{y'}_i \in \mathbb{R}^m$ of an image $\boldsymbol{x}_i$ for a novel task. Subsequently, $\boldsymbol{e}_k, \boldsymbol{y}_i$, and $\boldsymbol{y'}_i$ are fed into each subspace block (see Fig. 2**(a)**), where they are projected onto a subspace $\mathcal{P}_j$, that is constructed using the base class features in the visual embedding space (see Fig. 3) such that the Euclidean distance between the visual and semantic features are minimized. Then, the $j$-th subspace block outputs novel projected representations $\hat{\boldsymbol{e}}_{kj}$, $\hat{\boldsymbol{y}}_{ij}$ for the semantic and visual features, respectively. Further details on the subspace and VAE blocks are given in Section 3.3 and 3.4, respectively. The proposed architecture can operate by varying the

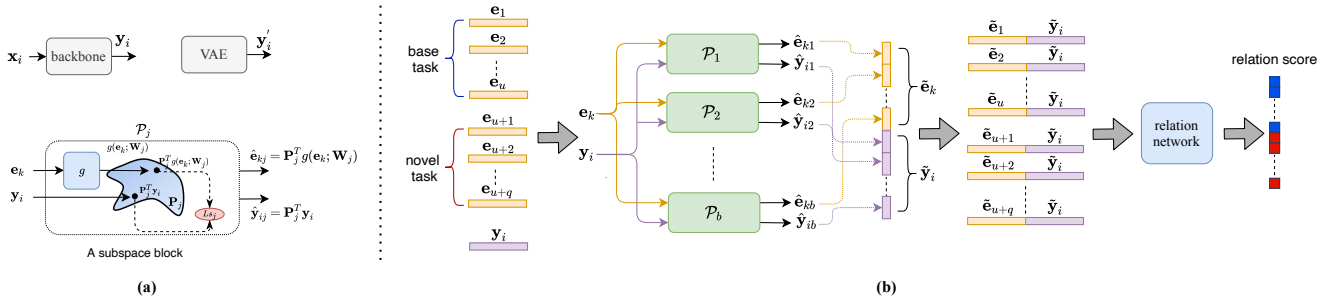



Figure 2: The proposed architecture. (a) Visual feature extraction block, which is a pre-trained CNN model, takes the input image $\mathbf{x}_i$ and outputs the feature vector $\boldsymbol{y_i}$. Also, we have a VAE module which generates visual features for novel tasks. A single subspace block $\mathcal{P}_j$ that takes $\boldsymbol{e_k}$ and $\boldsymbol{y_i}$ as the inputs and generates new feature representations $\tilde{\boldsymbol{e}}_{kj}$ and $\tilde{\boldsymbol{y}}_{ij}$. $g(\boldsymbol{e}_k; \boldsymbol{W}_j)$ is the projection of the $\boldsymbol{e}_k$, and $\boldsymbol{y}_i$ are projected onto subspace $\boldsymbol{P}_j$. (b) Overall architecture, where the output of all subspace blocks are concatenated in order to generate a new richer representation $\tilde{\boldsymbol{e}_k}$ and $\tilde{\boldsymbol{y}_i}$. For a given visual feature, $\boldsymbol{y_i}$, we forward semantic word vectors of both base and novel classes to multiple subspaces $\boldsymbol{P}_1, \boldsymbol{P}_2 \ldots \boldsymbol{P}_b$ generating a richer representation of both visual and semantics. Finally, a relation network compares each visual-semantic pair to estimate the final prediction scores.

number of subspaces, where the optimal number is found via cross validation. Once a collection of $\hat{\boldsymbol{y}}_{ij}$ and $\hat{\boldsymbol{e}}_{kj}$ has been extracted from a mixture of subspaces, they are rearranged and concatenated into a pair of representations $\tilde{\boldsymbol{y}}_i$ and $\tilde{\boldsymbol{e}}_k$ (see Fig.2 **(b)**). Following the idea in [8], for every training session (both base and incremental), we store a prototype $\hat{\boldsymbol{y}}^{\mathcal{M}}$ which is the average of all available visual feature representations for each class in the memory $\mathcal{M}$. For a novel task, we forward all semantic embeddings associated with the base and novel tasks to the subspace blocks for the corresponding visual features (both real and synthesized features $\{\boldsymbol{y}_i, \boldsymbol{y'}_i\}$). Finally, we forward $\tilde{\boldsymbol{y}}_i$ and $\tilde{\boldsymbol{e}}_k$ into the relation network [34] that ultimately predicts the label of the input by comparing visual and semantic alignment (see Fig. 2 **(b)**).

### 3.3. Subspace Projection

Modeling data by projection onto subspaces has been widely used in many computer vision and machine learning applications [22, 1, 17, 16, 32, 23]. Our model learns neighborhood embeddings in a low-dimensional space such that the visual and semantic features can be projected onto subspaces while preserving locality relationships. The basis vectors of the subspaces remain fixed during training. This strategy reduces overfitting of learning in a limited-data regime. Furthermore, the structure of the embedding space is preserved for the first task, where there are many training instances of each category.

**Subspace Generation Procedure:** Subspaces are generated based on visual feature instances of the first task $\mathcal{Q}^1$. The visual feature $\boldsymbol{y}_i$ is extracted from the pre-trained backbone that is trained on the samples of the first task. We use the $k$-means clustering method to partition the visual features into $b$ groups based on similarity between features (*e.g.*, the cosine distance). Specifically, the $j^{\text{th}}$ cluster composed of $\mathcal{N}_j$ samples is defined as $\mathcal{K}_j = \{\boldsymbol{y}_i\}_{i=1}^{\mathcal{N}_j}$. In creating subspaces from the samples within a cluster, we empirically observe that the Singular Value Decomposition (SVD) performs reasonably well in our setup. We decompose the matrix consisting of samples within a cluster as $\mathcal{K}_j = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$. Then, the $n$ leading left singular vectors $\boldsymbol{U}$ form an orthogonal basis for the $j^{\text{th}}$ subspace which we denote by $\boldsymbol{P}_j$, *i.e.*, $\mathbb{R}^{m\times n} \ni \boldsymbol{P}_j = [\boldsymbol{p}_1, ..., \boldsymbol{p}_n]\,;\ \boldsymbol{P}_j^\top \boldsymbol{P}_j = \mathbf{I}_n$. As an illustration, we employ the subspace method on the CUB dataset [37] (see Fig. 3).

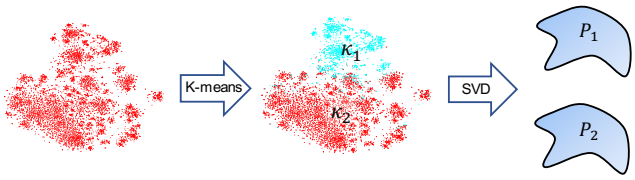



Figure 3: A toy example of the subspace generation procedure on the CUB dataset [37]. In the CUB dataset, the first task, base, consists of 100 classes (red points shown to the left). First, we apply $k$-means clustering to form two clusters $\mathcal{K}_1$ and $\mathcal{K}_2$ (shown in cyan and red in the middle). Then, SVD is applied to generate subspaces $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$.

**Subspace Block:** The inputs to the subspace block are semantic and visual features, and the outputs are the projected semantic and visual feature embeddings on a subspace as shown in Fig. 2 **(a)**. The network $g(\cdot)$, which consists of a lightweight fully connected network, is trained such that the Euclidean distance between the projected vectors is minimized, as illustrated in Fig 4. After training the block, new representations for the semantic and visual domains are generated as $\hat{\boldsymbol{e}}_{kj} = \boldsymbol{P}_j^\top g(\boldsymbol{e}_k, \boldsymbol{W}_j)$ and $\hat{\boldsymbol{y}}_{ij} = \boldsymbol{P}_j^\top \boldsymbol{y}_i$, respectively. The detailed steps in the subspace block are explained in Algorithm 1.

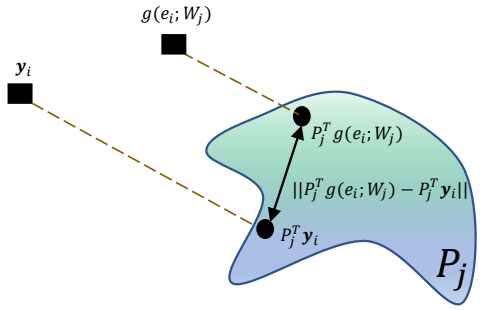



Figure 4: A geometrical interpretation of the loss used in the subspace block. A transformation of the semantic space is learnt such that the Euclidean distance between projected visual features and projected semantic vectors is minimized.

**Subspace Feature Embedding:** In order to obtain more expressive representations for semantic and visual cues, we utilize $b$ subspace blocks. Then, $\boldsymbol{e}_k$ and $\boldsymbol{y}_i$ are mapped into a new feature embedding for both the semantic feature $\tilde{\boldsymbol{e}}_k = \mathcal{C}\big(\boldsymbol{P}_1^T g(\boldsymbol{e}_k; \boldsymbol{W}_1), ..., \boldsymbol{P}_b^T g(\boldsymbol{e}_k; \boldsymbol{W}_b)\big)$ and the visual feature $\tilde{\boldsymbol{y}}_i = \mathcal{C}\big(\boldsymbol{P}_1^T \boldsymbol{y}_i, ..., \boldsymbol{P}_b^T \boldsymbol{y}_i\big)$, where $\mathcal{C}(\cdot)$ is the concatenation operator.

It should be noted that $\tilde{\boldsymbol{e}}_k$ and $\tilde{\boldsymbol{y}}_i$ represent the responses from a mixture of subspaces, where each subspace describes a cluster of similar visual instances (possibly indicating a superclass). Suppose that the superclass 'vehicle' may represent the subclasses *e.g.* 'car', 'bike', and 'bus'. If a visual instance of 'apple' comes as input, the computed $\tilde{\boldsymbol{e}}_k$ and $\tilde{\boldsymbol{y}}_i$ will get a lower response from the subspace of the 'vehicle' superclass compared to the subspace of the 'food' superclass. This intuition is different from previous work [41] where they consider multiple learnable embeddings, but there is no embedding representing the visual structure (*i.e.*, superclass) of the dataset. In contrast to the work in [41], our subspaces are semantically more meaningful. Moreover, the same subspace ($\boldsymbol{P}_j$), holding meaningful cluster information, is used across every training session to implicitly prevent forgetting for previously learned tasks and overfitting on a few examples when adapting novel classes. In this way, both visual and semantic vectors find rich representations considering both positive and negative superclass information from the mixture of subspaces.

**Projection onto Subspaces to Improve Generalization:** The aim of using subspaces in our method is to improve the generalization capability of the model. The assumption to achieve generalization is that the concepts of the base task share some similarity and allow transfer to the concepts of novel tasks. Conceptually, the set of visual features populates a small fraction of the space, and this characteristic inspires our approach by constructing multiple subspaces as low-dimensional and shared spaces for novel tasks. In essence, we conjecture that combining measurements on multiple subspaces induces regularization and generalization for learning novel tasks. In the FSCIL problem, maximizing similarity of features with similar concepts on a number of subspaces prevent the model to forget the previously learned concepts, and simultaneously reduce overfitting when learning from a few samples of novel classes. Furthermore, in our algorithm, the multi-modalities are analogous after projecting the features onto subspaces. Specifically, the outputs of a universal visual feature extractor (*e.g.*, ResNet [12]) are aligned to the semantic features on the shared subspace. As a result of the joint space between visual and semantic cues, the model becomes more generalizable across modalities and novel concepts.

**Algorithm 1** The proposed method for subspace block generation

**Input:** $\mathcal{Q}^1$
**Output:** $b$ subspace blocks
$\{\boldsymbol{y}_i\}_{i=1}^{N_1} \leftarrow$ extract visual features from a pre-trained network given $\{\boldsymbol{x}_i^1\}_{i=1}^{N_1}$
**Subspace generation**
$\mathcal{K}_j = \{\boldsymbol{y}_i\}_{i=1}^{\mathcal{N}_j},\ j = 1, ..., b \leftarrow$ construct $b$ clusters using $k$-means with visual feature data $\boldsymbol{y}$
$\boldsymbol{P}_j = [\boldsymbol{p}_1, ..., \boldsymbol{p}_n]\,,\ j = 1, ..., b \leftarrow$ generate $b$ subspaces using the SVD algorithm by using $\mathcal{K}_j = \{\boldsymbol{y}_i\}_{i=1}^{\mathcal{N}_j},\ j = 1, ..., b$, where $\mathcal{K}_j = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$ be the SVD of $\mathcal{K}_j$. Then, the $n$ leading left singular vectors of $\mathcal{K}_j$, captured by the first $n$ columns of $\boldsymbol{U}$ form an orthogonal basis for the $j-$th subspace which we denote by $\boldsymbol{P}_j$
**Subspace block initialization**
Initialize $b$ subspace blocks $\mathcal{P}_j, j = 1, ..., b$ with subspaces $\boldsymbol{P}_j, j = 1, ..., b$
**Return** $b$ subspace blocks

### 3.4. Synthesized Feature Generation

In this section, we discuss our synthetic feature generation process. We begin with a brief overview of the VAE and then discuss its adaptation to our pipeline. VAEs are a popular class of generative models that can be optimized end-to-end with gradient-based optimization techniques. VAEs comprise the ability to model complex distributions, starting from a simple prior, and are ubiquitously used in various modern applications.

In its vanilla form, a VAE consists of an *encoder* and a *decoder*, which are typically modeled using neural networks. The goal of the encoder is to model an approximate posterior distribution $q(\mathbf{z}|\mathbf{y}) \approx p(\mathbf{z}|\mathbf{y})$, where $\mathbf{y}$ and $\mathbf{z}$ are the feature and latent distributions, respectively. A critical assumption used in VAEs is that $p(\mathbf{z}|\mathbf{y})$ is a Gaussian distribution. Therefore, the encoder outputs the parameters—mean and standard deviation—per feature $\mathbf{y}$, which is then used

to construct the approximate posterior $q(\mathbf{z}|\mathbf{y})$. Similarly, the decoder aims to model the distribution $p(\mathbf{y}|\mathbf{z})$, given an input $\mathbf{z} \sim q(\mathbf{z}|\mathbf{y})$. The training objective of the VAE is to maximize the data likelihood,

$$\log p(\mathbf{y}) = ELBO + \mathbb{KL}[q(\mathbf{z}|\mathbf{y})||p(\mathbf{z}|\mathbf{y})], \quad (1)$$

where $ELBO$ is the *evidence-lower-bound* defined as,

$$ELBO = \mathbb{E}_{z\sim q(\mathbf{z}|\mathbf{y})}[\log p(\mathbf{y}|\mathbf{z})] - \mathbb{KL}[q(\mathbf{z}|\mathbf{y})||p(\mathbf{z})], \quad (2)$$

and $\mathbb{KL}(\cdot||\cdot)$ is the KL-divergence. Note that the KL divergence is always non-negative, hence, maximizing $ELBO$ is equivalent to maximizing the data likelihood. In practice, we minimize the negative $ELBO$ and our loss function becomes,

$$L_{\text{VAE}} = -\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{y})}[\log p(\mathbf{y}|\mathbf{z})] + \mathbb{KL}[q(\mathbf{z}|\mathbf{y})||p(\mathbf{z})], \quad (3)$$

where $p(\mathbf{z})$ is a standard normal distribution. Consider small sets of features $\{\mathbf{y}_c\}$ for task $t$, extracted from a pre-trained network, per class $c \in \mathcal{L}^t$. Our aim is to learn a VAE that can model the true feature distribution of each class $c$. To this end, we maximize the $ELBO$ of $\log(p(\mathbf{y}_c))$ during the training by utilizing the loss in Eq. 3. At inference time, we randomly input a feature $\mathbf{y} \sim \{\mathbf{y}_c\}$ from each class to the encoder to obtain the approximate posterior $q(\mathbf{z}|\mathbf{y})$. The latent codes are then sampled from $q(\mathbf{z}|\mathbf{y})$ and fed to the decoder to obtain synthetic features that belong to the corresponding class $c$.

We generate synthesized features for novel classes using the described VAE model at each incremental session. We train our proposed FSCIL model by augmenting the generated features with a few available novel class instances. It balances the number of instances used to train each task. Consequently, the training does not get biased to the classes of any session, i.e., reduced overfitting, especially towards novel classes. It is important to note that the VAE *does not* contain the knowledge of the entire dataset. In other words, the VAE only accesses the training samples of the current task to generate synthetic features. Furthermore, The VAE is not frozen for novel tasks, *i.e.,* we fine-tune it for novel classes.

### 3.5. Training and Inference

To train our model for the task $t$, we forward all the training samples $\{\mathbf{x}_i\}_{i=1}^{N_t}$ of the current task $\mathcal{Q}^t$ to the backbone to extract a set of visual representations $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^{N_t}$. Also, for all previous tasks, we store one prototype $\boldsymbol{y}_c^{\mathcal{M}}$ per class $c$, which is the average of all available visual feature representations for each class, in a small memory $\mathcal{M}$.

In the proposed architecture, two loss functions are utilized for end-to-end training of the model. The loss function

**Algorithm 2** The proposed method for FSCIL

**Input:** $\mathcal{Q} = \{\mathcal{Q}^1, ..., \mathcal{Q}^T\}$
**Output:** A trained model to find $l^*$ for all $\boldsymbol{x}^c$, where $c \in \bigcup_{t=1}^{T} \mathcal{L}^t$
$\mathcal{M} \leftarrow \{\}$
$\{\boldsymbol{y}_i^1\}_{i=1}^{N_1} \leftarrow$ extract visual features from a pre-trained network given $\{\boldsymbol{x}_i^1\}_{i=1}^{N_1}$
$b$ subspace blocks $\leftarrow$ apply algorithm 1
**for** $t = 1$ to $T$ **do**
    Gen. synthesized features, $\mathbf{y}'$, using a VAE module
    **repeat**
        **for** $\forall \boldsymbol{I}$ in $\boldsymbol{y} \cup \mathbf{y}' \cup \mathcal{M}$ **do**
            Forward visual features $\boldsymbol{y}_i^t$ and semantic representation $e_k \in \mathcal{E}^t$
            Calculate the loss using Eq 8
            Backpropagate and update $\boldsymbol{W}_j$ and $\theta$
    **until** convergence
    $\mathcal{M} \leftarrow$ UPDATEMEMORY$(\mathcal{Q}^t, \mathcal{M}, \mathcal{L}^t)$
**function** UPDATEMEMORY$(\mathcal{Q}^t, \mathcal{M}, \mathcal{L}^t)$
    **for** $c = 1$ to $\mathcal{L}^t$ **do**
        Calculate a prototype $\mathbf{y}_c^{\mathcal{M}}$ for each class by averaging of all training samples from each class
        $\mathcal{M} \leftarrow \mathcal{M} \cup (\mathbf{y}_c^{\mathcal{M}}, \mathbf{l}_c^t)$
    **return** $\mathcal{M}$

for optimizing subspace blocks is defined as,

$$L_p = \frac{1}{bK}\sum_{j=1}^{b}\sum_{\mathbf{y}_i\in\mathcal{S}} \left\|\boldsymbol{P}_j^\top \boldsymbol{y}_i - \boldsymbol{P}_j^\top g(\boldsymbol{e}_i; \boldsymbol{W}_j)\right\|_2^2, \quad (4)$$

where $\mathcal{S} = \mathcal{Y} \cup \mathcal{M}$ and $K$ is the number of training samples in $\mathcal{S}$. The above loss function forces the model to learn the necessary transformation applied to the semantic vectors, *i.e*, this loss function minimizes the Euclidean distance between the projected feature and semantic vectors.

Moreover, the new embedding of semantic $\tilde{\boldsymbol{e}}_k$ and visual $\tilde{\boldsymbol{y}}_i$ features obtained from the set of subspace blocks are concatenated and fed into a relation module [34], which produces a score in the range $[0, 1]$, indicating the level of similarity between $\tilde{\boldsymbol{y}}_i$ and $\tilde{\boldsymbol{e}}_k$. We generate this score for each of the classes in both the current task and previous tasks, which is defined as,

$$R_{ik} = r(\mathcal{C}(\tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{e}}_k); \theta),\ k \in \mathcal{L}_{tl}, \quad (5)$$

where $\mathcal{L}_{tl} = \bigcup_{i=1}^{t} \mathcal{L}^i$. Finally, we apply a binary cross entropy loss to train the model as,

$$L_{cls} = -\frac{1}{MK}\sum_{k\in\mathcal{L}_{tl}}\sum_{\mathbf{y}_i\in\mathcal{S}}\Big(\mathbf{1}(l_i^t == k)\log(R_{ik}) + \big(1 - \mathbf{1}(l_i^t == k)\big)\log(1 - R_{ik})\Big), \quad (6)$$

where $M$ is the number classes in $\mathcal{L}_{tl}$. The total loss for real features is denoted as,

$$L_r = L_{cls} + L_p. \tag{7}$$

Additionally, we have a separate loss function for synthetic features $L_s$, similar to Eq. 4 and Eq. 6. Finally, we combine the loss functions of the real and synthetic features as,

$$L_t = \alpha L_r + (1 - \alpha) L_s, \tag{8}$$

where $\alpha$ is an empirically chosen hyper-parameter. At inference time, given the trained subspaces, the relation module and an unlabeled sample $\boldsymbol{x}^c$, $c \in \mathcal{L}_{tl}$, the prediction of the label is determined by

$$l^* = \underset{k \in \mathcal{L}_{tl}}{\arg\max}\ r(\mathcal{C}(\tilde{\boldsymbol{y}}^c, \tilde{\boldsymbol{e}}_k); \theta)\ . \tag{9}$$

The overall training process is described in Algorithm 2.

## 4. Experiments

**Datasets:** In this paper, we utilise three datasets, CUB200 [38], MiniImageNet [36], and CIFAR100 [19], to assess our proposed method. CUB200 consists of 200 classes, divided into 6000 training and 6000 testing instances, where the image size is 224 × 224. MiniImageNet consists of 100 classes, including 500 training instances and 100 testing instances. Likewise, CIFAR100 comprises 100 classes, where each class includes 500 training samples and 100 testing samples. In this work, we use the setting introduced by [35]. In the CUB200 dataset, 100 classes are selected as the base classes, and the remaining 100 classes are split into ten sessions, where a 10-way 5-shot setting is considered. For CIFAR100 and MiniImageNet, 60 classes are chosen as the base, and the 40 classes are considered the novel set, where they are split into eight novel sessions.

**Semantic Features:** We employ unsupervised word vectors acquired from the unannotated text corpus as a class semantic embedding. For CUB200, MiniImageNet, and CIFAR100, we used 400, 1000, and 300 dimensional word2vec [21], respectively.

**Evaluation:** In all experiments, we use top-1 accuracy to evaluate the methods, where the predicted label is compared against the ground truth label as the successful prediction.

**Hyperparameters:** To find hyperparameters, we conducted a grid search. We split the training set into two sets: a base set, which consists 60% of the training classes, and a validation set which consists of the rest of the classes added incrementally. The hyperparameters $b$, $n$, and $\alpha$ are set to 3, 256, and 0.6 for CUB200, 5, 256, and 0.65 for CIFAR100, 5 and 256, and 0.55 for MiniImageNet.

**Implementation details**[1]**:** For obtaining visual features, we used ResNet-18 [12], where visual features are extracted from the last pooling layer with 512 dimensions. The backbone is trained on the base task and kept frozen for the coming tasks. For the subspace block, we use two fully-connected layers with 1200 and 2048 hidden units, respectively, with ReLU as the non-linear function, denoted by $g$ in Figure 2. For the relation module, we use three fully-connected layers with 2048, 1024, and 1 hidden unit, where the first two layers have a ReLU function, and the last layer has a Sigmoid function. For training the above networks, we use the Adam optimizer [18], where the learning rate and batch size were set to 0.0001 and 64, respectively.

Furthermore, we implement both the decoder and the encoder in the VAE as fully connected neural networks. Each network comprises three layers with 256 hidden units, and the dimension of our latent codes are 16. We use ReLu as the activation function for all the layers except for the last layer in the decoder. For training the VAE, We use the Adam optimizer with a learning rate of 0.01, and a batch size of 4. All the values were chosen empirically.

[1]Code is available at: https://git.io/JRb81

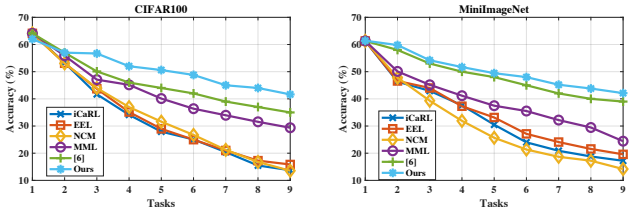



Figure 5: Results on ***(left)*** CIFAR100 and ***(right)*** MiniImageNet using the ResNet-18 architecture on the 5-way 5-shot FSCIL.

### 4.1. Main results

Here, we compare our proposed approach with state-of-the-art [26, 2, 13, 35, 8] on three well-known datasets, CUB200 [38], MiniImageNet [36], and CIFAR100 [19].

**CUB200 results:** We report the performance on the CUB200 dataset in Table 1. As can be seen, our proposed approach outperforms the state-of-the-art by a large margin ($> 10\%$ ) in the last task.

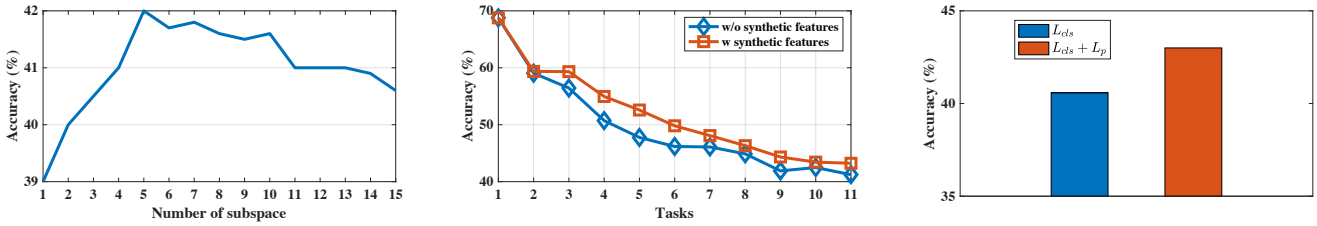
**CIFAR100 results:** We show the accuracy of our method on CIFAR100 dataset in Fig 5***(left)***. However, while our accuracy on the first task was almost 2% below the other methods, we still achieve better performance than the state-of-the-art by a large margin.

**MiniImageNet results:** Similar to other datasets, in Fig 5***(right)***, we beat state-of-the-art methods across all incremental tasks on MiniImageNet.

Unlike other methods, we achieve the best performance without using the traditional use of knowledge distillation techniques. The use of knowledge distillation methods for FSCIL may face several problems as discussed in [35]. For example, balancing the contribution between cross-entropy (CE) and KD losses leads to an unsatisfactory performance

| Method | Tasks/Sessions | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| iCaRL [26] | 68.68 | 52.65 | 48.61 | 44.16 | 36.62 | 29.52 | 27.83 | 26.26 | 24.01 | 23.89 | 21.16 |
| EEIL [2] | 68.68 | 53.63 | 47.91 | 44.20 | 36.30 | 27.46 | 25.93 | 24.70 | 23.95 | 24.13 | 22.11 |
| NCM [13] | 68.68 | 57.12 | 44.21 | 28.78 | 26.71 | 25.66 | 24.62 | 21.52 | 20.12 | 20.06 | 19.87 |
| AL-MML [35] | 68.68 | **62.49** | 54.81 | 49.99 | 45.25 | 41.40 | 38.35 | 35.36 | 32.22 | 28.31 | 26.28 |
| Cheraghian *et.al* [8] | 68.23 | 60.45 | 55.70 | 50.45 | 45.72 | 42.90 | 40.89 | 38.77 | 36.51 | 34.87 | 32.96 |
| Ours | 68.78 | 59.37 | **59.32** | **54.96** | **52.58** | **49.81** | **48.09** | **46.32** | **44.33** | **43.43** | **43.23** |

Table 1: CUB200 results with ResNet18 based on the 10-way 5-shot setting.




(a) (b) (c)

Figure 6: The impact of using (a) different number of subspaces, (b) synthesized features, and (c) loss functions in our proposed method.

trade-off. Moreover, learning new few-shot classes requires a higher learning rate to minimize CE. It can cause instability of the output logits that makes it difficult to minimize KD. The consistent performances of our approach result from the utilization of a subspace mixture and synthesized features of the novel class. The subspace mixture stores the old knowledge in such a way that the network does not catastrophically forget past training and synthesized feature does not overfit towards the novel class.

### 4.2. Ablation study

**The impact of subspaces:** Here, we evaluate the effect of subspace blocks in our proposed method in Fig. 6(a). We vary the number of subspaces by using different values of $k \in \{1, 2, \dots 15\}$ while applying k-means clustering. $k = 1$ means using a single global subspace, $b = 1$, while not capturing the superclass structure of visual similarity and dissimilarity that reside in the dataset. $k > 1$ means using a mixture of multiple subspaces $b > 1$, which captures the superclass cluster structure. We achieve the best result while using $b = 5$ using the MiniImageNet dataset where both the global and local structure attains a perfect balance. If $b$ is low or high, either global or local information dominates, respectively, making the system imbalanced.

**The impact of synthesized features:** Fig. 6 (b) shows the effect using synthesized features on the CUB200 dataset. One can notice that in almost all incremental training sessions, the results get improved while considering synthesized features. It tells us that augmentation of synthesized features brings extra knowledge for the novel classes and helps to not overfit towards a few real examples.

**The impact of different loss functions:** Fig. 6 (c) demonstrates the effect of using a classification loss $L_{cls}$ and a subspace learning loss $L_p$ on the CUB dataset. We notice using both losses, i.e., $L_{cls} + L_p$ works better than using only $L_{cls}$. The reason is that $L_p$ aligns visual features and semantic word vectors conditioning on a particular subspace. The aligned version of visual-semantics supports learning the relation network in the later stage better than the non-aligned version. Note that in both cases, we consider synthesized features during each incremental session. We do not use $L_p$ alone because the relation network cannot be learned without the classification loss, $L_{cls}$.

## 5. Conclusion

This paper proposes a mixture of subspaces-based method that works on real and synthesized visual features to address the FSCIL task. Traditional approaches of FSCIL struggle in catastrophic forgetting of base classes and overfit to novel class examples. Our proposed method minimizes those problems by constructing a mixture of subspaces and a VAE model for synthesized feature generation. Different subspaces capture various aspects of the visual cluster structure. Later, a mixture of individual subspaces represents features and semantics such that irrespective of a base and novel feature as input, our method can produce balanced predictions across all incremental sessions, which helps in the FSCIL tasks. Moreover, the VAE model augments synthesized features while learning novel classes with few-shot examples that mainly help to adapt incremental knowledge. We have experimented on three 2D image datasets and reported satisfactory results to demonstrate our contributions.

## References


[1] Ronen Basri and David W Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (2):218–233, 2003.

[2] Francisco M. Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-End Incremental Learning. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *ECCV 2018 - European Conference on Computer Vision*, Munich, Germany, Sept. 2018.

[3] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[4] Kuilin Chen and Chi-Guhn Lee. Incremental few-shot learning via vector quantization in deep embedded space. In *International Conference on Learning Representations*, 2021.

[5] Ali Cheraghian, Shafin Rahman, Dylan Campbell, and Lars Petersson. Mitigating the hubness problem for zero-shot learning of 3d objects. In *British Machine Vision Conference (BMVC'19)*, 2019.

[6] Ali Cheraghian, Shafin Rahman, Dylan Campbell, and Lars Petersson. Transductive zero-shot learning for 3d point cloud classification. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 912–922, 2020.

[7] Ali Cheraghian, Shafinn Rahman, Townim F Chowdhury, Dylan Campbell, and Lars Petersson. Zero-shot learning on 3d point cloud objects and beyond. *arXiv preprint arXiv:2104.04980*, 2021.

[8] Ali Cheraghian, Shafin Rahman, Pengfei Fang, Soumava Kumar Roy, Lars Petersson, and Mehrtash Harandi. Semantic-aware knowledge distillation for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[9] J. Dong, X. Li, and C. G. M. Snoek. Predicting visual features from text for image and video caption retrieval. *IEEE Transactions on Multimedia*, 20(12):3377–3388, 2018.

[10] Pengfei Fang, Mehrtash Harandi, and Lars Petersson. Kernel methods in hyperbolic spaces. In *Proceedings of the International Conference on Computer Vision*, 2021.

[11] S. Gidaris and N. Komodakis. Dynamic Few-Shot Visual Learning Without Forgetting. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[13] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a Unified Classifier Incrementally via Rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[14] Hexiang Hu, Wei-Lun Chao, and Fei Sha. Learning answer embeddings for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5428–5436, 2018.

[15] He Huang, Changhu Wang, Philip S. Yu, and Chang-Dong Wang. Generative dual adversarial network for generalized zero-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] Pan Ji, Mathieu Salzmann, and Hongdong Li. Efficient dense subspace clustering. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2014.

[17] Pan Ji, Mathieu Salzmann, and Hongdong Li. Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research).

[20] Pratik Mazumder, Pravendra Singh, and Piyush Rai. Large Scale Incremental Learning. In *AAAI*, 2021.

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119. 2013.

[22] Hiroshi Murase and Shree K Nayar. Visual learning and recognition of 3-d objects from appearance. *International journal of computer vision*, 14(1):5–24, 1995.

[23] I. Naseem, R. Togneri, and M. Bennamoun. Linear regression for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(11):2106–2112, Nov 2010.

[24] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational Continual Learning. In *International Conference on Learning Representations*, 2018.

[25] Shafin Rahman, Salman Khan, Nick Barnes, and Fahad Shahbaz Khan. Any-shot object detection. In Hiroshi Ishikawa, Cheng-Lin Liu, Tomas Pajdla, and Jianbo Shi, editors, *Computer Vision – ACCV 2020*, pages 89–106, Cham, 2021. Springer International Publishing.

[26] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental Classifier and Representation Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, 2017.

[27] Mengye Ren, Renjie Liao, Ethan Fetaya, and Richard S. Zemel. Incremental Few-Shot Learning with Attention Attractor Networks, booktitle= Advances in Neural Information Processing Systems (NeurIPS), year = 2019,.

[28] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, , and Gerald Tesauro. Learning to Learn Without Forgetting By Maximizing Transfer and Minimizing Interference. In *International Conference on Learning Representations*, 2019.

[29] Mert Bulent Sariyildiz and Ramazan Gokberk Cinbis. Gradient matching generative networks for zero-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[30] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8247–8255, 2019.

[31] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2990–2999. Curran Associates, Inc., 2017.

[32] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4136–4145, 2020.

[33] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc., 2017.

[34] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, June 2018.

[35] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-Shot Class-Incremental Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[36] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3630–3638. Curran Associates, Inc., 2016.

[37] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[38] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[39] Jiamin Wu, Tianzhu Zhang, Zheng-Jun Zha, Jiebo Luo, Yongdong Zhang, and Feng Wu. Self-supervised domain-aware generative network for generalized zero-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[40] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large Scale Incremental Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[41] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *CVPR*, June 2016.

[42] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5542–5551, 2018.

[43] Yongqin Xian, Saurabh Sharma, Bernt Schiele, and Zeynep Akata. F-vaegan-d2: A feature generating framework for any-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[44] Yongqin Xian, Saurabh Sharma, Bernt Schiele, and Zeynep Akata. f-vaegan-d2: A feature generating framework for any-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10275–10284, 2019.

[45] Meng Ye and Yuhong Guo. Progressive ensemble networks for zero-shot recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11728–11736, 2019.

[46] Hadi Zanddizari, Nam Nguyen, Behnam Zeinali, and J Morris Chang. A new preprocessing approach to improve the performance of cnn-based skin lesion classification. *Medical & Biological Engineering & Computing*, 59(5):1123–1131, 2021.

[47] Behnam Zeinali, Di Zhuang, and J Morris Chang. Esai: Efficient split artificial intelligence via early exiting using neural architecture search. *arXiv preprint arXiv:2106.12549*, 2021.

[48] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[49] Chenchen Zhu, Fangyi Chen, Uzair Ahmed, Zhiqiang Shen, and Marios Savvides. Semantic relation reasoning for shot-stable few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8782–8791, 2021.