iNeMo: Incremental Neural Mesh Models for Robust Class-Incremental Learning

Tom Fischer[1], Yaoyao Liu[2], Artur Jesslen[3], Noor Ahmed[1], Prakhar Kaushik[2], Angtian Wang[2], Alan L. Yuille[2], Adam Kortylewski[3,4], and Eddy Ilg[1]

[1] Saarland University, Saarbrücken, Germany
{fischer, ilg}@cs.uni-saarland.de
[2] Johns Hopkins University, Baltimore, USA
{yliu538, angtianwang, ayuille1}@jhu.edu
[3] University of Freiburg, Freiburg, Germany
{jesslen, kortylew}@cs.uni-freiburg.de
[4] Max Planck Institute for Informatics, Saarbrücken, Germany
akortyle@mpi-inf.mpg.de

Abstract. Different from human nature, it is still common practice today for vision tasks to train deep learning models only initially and on fixed datasets. A variety of approaches have recently addressed handling continual data streams. However, extending these methods to manage out-of-distribution (OOD) scenarios has not effectively been investigated. On the other hand, it has recently been shown that non-continual neural mesh models exhibit strong performance in generalizing to such OOD scenarios. To leverage this decisive property in a continual learning setting, we propose incremental neural mesh models that can be extended with new meshes over time. In addition, we present a latent space initialization strategy that enables us to allocate feature space for future unseen classes in advance and a positional regularization term that forces the features of the different classes to consistently stay in respective latent space regions. We demonstrate the effectiveness of our method through extensive experiments on the Pascal3D and ObjectNet3D datasets and show that our approach outperforms the baselines for classification by $2-6\%$ in the in-domain and by $6-50\%$ in the OOD setting. Our work also presents the first incremental learning approach for pose estimation. Our code and model can be found at github.com/Fischer-Tom/iNeMo.

Keywords: Class-incremental learning · 3D pose estimation

1 Introduction

Humans inherently learn in an incremental manner, acquiring new concepts over time, with little to no forgetting of previous ones. In contrast, trying to mimic the same behavior with machine learning suffers from *catastrophic forgetting* [21, 36, 37], where learning from a continual stream of data can destroy the knowledge that was previously acquired. In this context, the problem was formalized as *class-incremental learning* and a variety of approaches

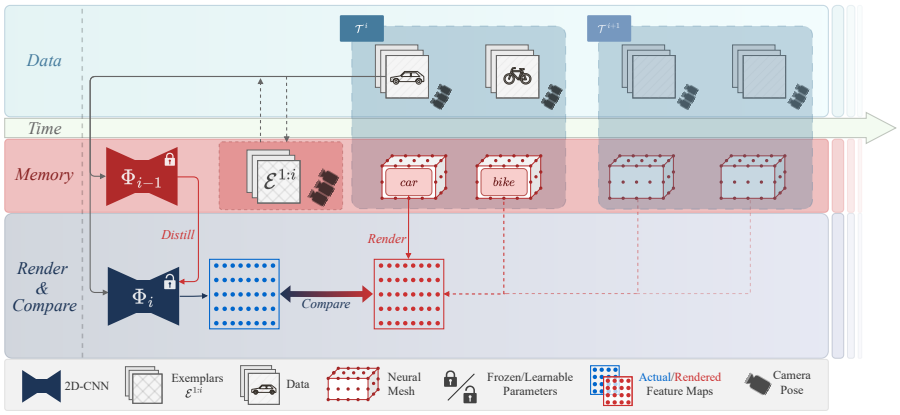

Fig. 1: We present iNeMo that can perform class-incremental learning for pose estimation and classification, and performs well in out-of-distribution scenarios. Our method receives tasks $\mathcal{T}^i$ over time that consist of images with camera poses for new classes. We build up on Neural Mesh Models (NeMo) [52] and abstract objects with simple cuboid 3D meshes, where each vertex carries a neural feature. The neural meshes are optimized together with a 2D feature extractor Φ_i and render-and-compare can then be used to perform pose estimation and classification. We introduce a memory that contains an old feature extractor Φ_{i-1} for distillation, a replay buffer $\mathcal{E}^{1:(i-1)}$ and a growing set of neural meshes $\mathfrak{N}$. Our results show that iNeMo outperforms all baselines for incremental learning and is significantly more robust than previous methods.

have been proposed to address catastrophic forgetting for models that work in-distribution [11,17,26,30,31,45]. However, extending these methods to effectively manage out-of-distribution (OOD) scenarios [64] to the best of our knowledge has not been investigated.

Neural mesh models [52] embed 3D object representations explicitly into neural network architectures, and exhibit strong performance in generalizing to such OOD scenarios for classification and 3D pose estimation. However, as they consist of a 2D feature extractor paired with a generative model, their extension to a continual setting with existing techniques is not straight forward. If one would only apply those techniques to the feature extractor, the previously learned neural meshes would become inconsistent and the performance of the model would drop.

In this paper, we therefore present a strategy to learn neural mesh models incrementally and refer to them as incremental Neural Mesh Models (iNeMo). As shown in Figure 1, in addition to the conventional techniques of knowledge distillation and maintaining a replay buffer, our approach introduces a memory that contains a continuously growing set of meshes that represent object categories. To establish the learning of the meshes in an incremental setting, we extend the contrastive learning from [52] by a latent space initialization strategy that enables us to allocate feature space for future unseen classes in advance, and

a positional regularization term that forces the features of the different classes to consistently stay in respective latent space regions. Through extensive evaluaitons on the Pascal3D [61] and ObjectNet3D [60] datasets, we demonstrate that our method outperforms existing continual learning techniques and furthermore surpasses them by a large margin for out-of-distribution samples. Overall, our work motivates future research on joint 3D object-centric representations. In summary, the contributions of our work are:

1. For the first time, we adapt the conventional continual learning techniques of knowledge distillation and replay to the 3D neural mesh setting.
2. We propose a novel architecture, that can grow by adding new meshes for object categories over time.
3. To effectively train the features of the meshes, we introduce a strategy to partition the latent space and maintain it when new tasks are integrated.
4. We demonstrate that incremental neural mesh models can outperform 2D baselines that use existing 2D continual learning techniques by $2-6\%$ in the in-domain and by $6-50\%$ in the OOD setting.
5. Finally, we introduce the first incremental approach for pose estimation and show that the neural mesh models outperform 2D baselines.

2 Related Work

2.1 Robust Image Classification and Pose Estimation

Image Classification has always been a cornerstone of computer vision. Groundbreaking models such as ResNets [13], Transformers [51], and Swin Transformers [32] have been specifically designed for this task. However, these models predominantly target the in-distribution setting, leading to a significant gap in performance when faced with challenging benchmarks that involve synthetic corruptions [14], occlusions [55], and out-of-distribution (OOD) images [64]. Attempts to close this performance gap have included data augmentation [15] and innovative architectural designs, such as the analysis-by-synthesis approach [22]. Along this line of research, recently neural mesh models emerged as a family of models [35, 52–54] that learn a 3D pose-conditioned model of neural features and predict 3D pose and object class [19] by minimizing the reconstruction error between the actual and rendered feature maps using render-and-compare. Such models have shown to be significantly more robust to occlusions and OOD data. However, they can so far only be trained on fixed datasets. In this work, we present the first approach to learn them in a class-incremental setting.

Object Pose Estimation has been approached primarily as a regression problem [39,50] or through keypoint detection and reprojection [66] in early methods. More recent research [18,25] addresses object pose estimation in complex scenarios like partial occlusion. NeMo [52] introduces render-and-compare techniques for category-level object pose estimation, showcasing enhanced robustness in OOD conditions. Later advancements in differentiable rendering [56] and data

augmentation [23] for NeMo have led to further improvements in robust category-level object pose estimation, achieving state-of-the-art performance. However, these approaches are confined to specific object categories and are designed for fixed training datasets only. In contrast, our method for the first time extends them to the class-incremental setting.

2.2 Class-Incremental Learning

Class-incremental learning (also known as continual learning [2, 10, 33] and lifelong learning [1, 8, 9]) aims at learning models from sequences of data. The foundational work of [6, 45] replays exemplary data from previously seen classes. The simple strategy has inspired successive works [7, 58]. However, for such methods, sampling strategies and concept drift can impact overall performance. As a mitigation, more recent methods [17, 59] combine replay with other notable regularization schemes like knowledge distillation [26]. In general, class-incremental methods leverage one or more principles from the following three categories: (1) exemplar replay methods build a reservoir of samples from old training rounds [4, 28, 31, 34, 43, 45, 47] and replay them in successive training phases as a way of recalling past knowledge, (2) regularization-based (distillation-based) methods try to preserve the knowledge captured in a previous version of the model by matching logits [26, 45], feature maps [11], or other information [20, 27, 42, 48, 49, 57] in the new model, and (3) network-architecture-based methods [29, 57] design incremental architectures by expanding the network capacity for new class data or freezing partial network parameters to retain the knowledge about old classes.

In our work, we make use of principles from all three of the above by leveraging a replay memory, presenting a novel regularization scheme and adding newly trained neural meshes to the model over time. To the best of our knowledge, our method is the first to combine a 3D inductive bias with these strategies.

3 Prerequisites

3.1 Class Incremental Learning (CIL)

Conventionally, classification models are trained on a single training dataset $\mathcal{T}$ that contains all classes. Multi-class incremental learning departs from this setting by training models on sequentially incoming datasets of new classes that are referred to as tasks $\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^{N_{task}}$, where each task may contain more than one new class. After training on a new task $\mathcal{T}^i$, the model may be evaluated on a test dataset $\mathcal{D}^{1:i}$ that contains classes from all tasks up to i.

When being trained on new tasks through a straightforward fine-tuning, models suffer from *catastrophic forgetting* [21], which leads to bad performance on the previously seen classes. An intuitive approach to mitigate this effect is to use a *replay buffer* [45] that stores a few examplars $|\mathcal{E}^i| \ll |\mathcal{T}^i|$ from previous tasks and includes them with training data of the new task. Another common

technique is *knowledge distillation* [16, 26] that keeps a copy of the model before training on the new task and ensures that distribution of the feature space from the old and new models are similar when presented the new data.

3.2 Neural Mesh Models

Neural mesh models combine a 2D feature extractor with generative 3D models, as shown by Wang et al. [52] in their Figure 1. The generative models are simple 3D abstractions in the form of cuboids for each class c that are represented as meshes $\mathfrak{N}_c = (\mathcal{V}_c, \mathcal{A}_c, \Theta_c)$, where $\mathcal{V}_c$ denotes the vertices, $\mathcal{A}_c$ denotes the triangles and Θ_c denotes the neural vertex features. The meshes are additionally accompanied by a set of background features $\mathcal{B}$. Given camera intrinsics and extrinsics, a mesh can then be rendered to a 2D feature map. The 2D feature extractor is usually a 2D CNN $\Phi(I)$ that takes the image as input to extract a feature map and is shared among all classes [19]. Render-and-compare can then be used to check if the features rendered from the mesh align with the features extracted from the image to perform pose estimation [52] or classification [19]. We denote a normalized feature vector at vertex k as θ_c^k, its visibility in the image as o_c^k, its projected integer image coordinates as $\pi_c(k)$, and $f_{\pi_c(k)}$ as the normalized feature vector from the 2D feature extractor that corresponds to the rendered vertex k.

During training, images and object poses are provided, and the vertex features Θ, background features $\mathcal{B}$, and the 2D feature extractor Φ are trained. We model the probability distribution of a feature f being generated from a vertex v_c^k by defining $P(f|\theta_c^k)$ using a von Mises-Fisher (vMF) distribution to express the likelihood:

$$P(f|\theta_c^k, \kappa) = C(\kappa) e^{\kappa(f^\top \cdot \theta_c^k)} \ , \tag{1}$$

with mean θ_c^k, concentration parameter κ, and normalization constant $C(\kappa)$ [52]. In the next step, the extracted feature $f_{\pi_c(k)}$ is inserted into $P(f|\theta_c^k, \kappa)$ and maximized using contrastive learning. Simultaneously, the likelihood of all other vertices and background features is minimized:

$$\max \qquad P(f_{\pi_c(k)}|\theta_c^k, \kappa), \tag{2}$$

$$\min \sum_{\theta^m \in \bar{\theta}_c^k} P(f_{\pi_c(k)}|\theta^m, \kappa), \tag{3}$$

where the alternative vertices are defined as $\bar{\theta}_c^k = \{\mathcal{B} \cup \Theta_{\bar{c}} \cup (\Theta_c \setminus \mathcal{N}_c^k)\}$ with the neighborhood $\mathcal{N}_c^k = \{\theta_i \,|\, \|v_i^k - v_c^k\| < R \wedge v_i^k \in \mathcal{V}_c \setminus v_k^c\}$ around v_c^k determined by some pre-defined distance threshold R. We formulate the Equations 2 and 3 into a single loss by taking the negative log-likelihood:

$$\mathcal{L}_{\text{train}} = -\sum_k o_c^k \cdot \log\left(\frac{e^{\kappa(f_{\pi_c(k)}^\top \cdot \theta_c^k)}}{\sum_{\theta_m \in \bar{\theta}_c^k} e^{\kappa(f_{\pi_c(k)}^\top \cdot \theta_m)}}\right), \tag{4}$$

where considering κ as a global hyperparameter allows cancelling out the normalization constants $C(\kappa)$.

The concentration parameter κ determines the spread of the distribution and can be interpreted as an inverse temperature parameter. In practice, the neural vertex features Θ and the background features $\mathcal{B}$ are unknown and need to be optimized jointly with the feature extractor Φ. This makes the training process initially ambiguous, where a good initialization of Φ and Θ is critical to avoid divergence. After each update of Φ, we therefore follow Bai *et al.* [3] and use the momentum update strategy to train the foreground model Θ_c of a class c, as well as the background model $\mathcal{B}$:

$$\theta_c^{k,new} \leftarrow o_c^k(1-\eta) \cdot f_{\pi_c(k)} + (1 - o_c^k + \eta \cdot o_c^k)\theta_c^k \ , \tag{5}$$

where η is the momentum parameter. The background model $\mathcal{B}$ is updated by sampling $N_{bgupdate}$ feature vectors at pixel positions that are not matched to any vertex of the mesh and replace the $N_{bgupdate}$ oldest features in $\mathcal{B}$. Both $N_{bgupdate}$ and η are hyperparameters. For a more detailed description of this process, we refer to the supplementary material.

4 Incremental Neural Mesh Models (iNeMo)

Our goal is to learn a model that generalizes robustly in OOD scenarios, while being capable of performing class-incremental learning. To achieve this, we build up on neural mesh models [52] and present a novel formulation for class-incremental learning for classification and object pose estimation that we call iNeMo. An overview is provided in Figure 1.

Challenges in CIL. In the non-incremental setting, the contrastive loss in Equation 4 does not explicitly enforce separating classes, although in practice it is observed that the classes are separated well and accurate classification can be achieved [19]. A naive extension of neural mesh models to class-incremental learning is to simply add a mesh $\mathfrak{N}_c$ for each new class. However, the challenge lies in updating the shared 2D feature extractor Φ. If adding classes naively, achieving a discriminative latent space requires restructuring it as a whole and therefore implies significant changes in both, the CNN backbone Φ and the neural meshes Θ, leading to catastrophic forgetting if no old training samples are available or other measures are taken. Therefore, in the following we present a novel class-incremental learning strategy that maintains a well structured latent space from the beginning.

4.1 Initialization

Latent Space. As the features θ_c^k are normalized, they lie on a unit sphere. We therefore define an initial population $\boldsymbol{H} = \{\boldsymbol{h}_j \,|\, \boldsymbol{h}_j \in \mathbb{R}^d \wedge \|\boldsymbol{h}_j\| = 1\}$ of the latent space for all vertices and classes by uniformly sampling the sphere. To partition the latent space, we define a fixed upper bound of classes N. We then generate centroids $\boldsymbol{E} = [\boldsymbol{e}_1, ..., \boldsymbol{e}_N]$ for all the classes on the unit sphere that are

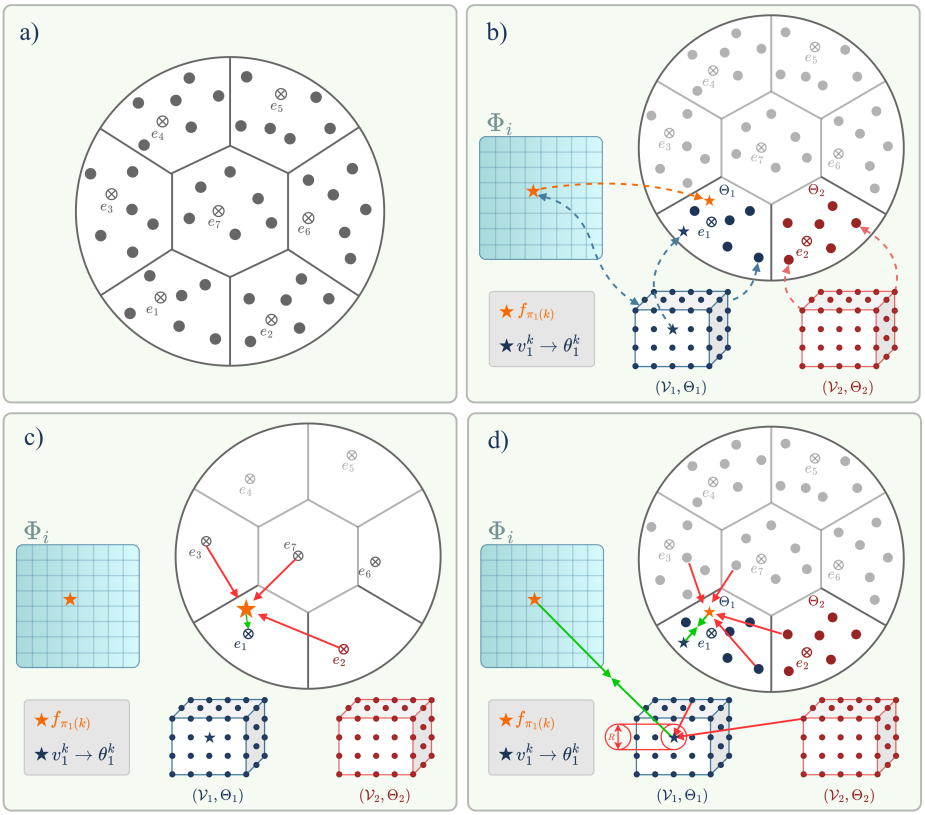

Fig. 2: Overview of Regularization: a) The features are constrained to lie on a unit sphere and the latent space is initially uniformly populated. Centroids e_i are then computed to lie maximally far apart, and the feature population is partitioned for a maximum number of classes. **b)** When starting a new task, the vertex features for each new cube from this task are randomly initialized from some class partition. By projecting the locations of the vertices to images, corresponding image features are determined as illustrated by the orange star. **c)** To avoid entanglement, we regularize the latent space by constraining the image feature to stay within the class partition using $\mathcal{L}_{etf}$. **d)** We then employ the contrastive loss $\mathcal{L}_{\text{cont}}$ that pulls the vertex and image features together and separates the image feature from other features of its own, and the other meshes.

pairwise maximally far apart by solving the equation for a simplex Equiangular Tight Frame (ETF) [40]:

$$\boldsymbol{E} = \sqrt{\frac{N}{N-1}}\boldsymbol{U}(\boldsymbol{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top) \ , \tag{6}$$

where $\boldsymbol{I}_N$ denotes the $n-$dimensional identity matrix, $\mathbf{1}_n$ is an all-ones vector, and $\boldsymbol{U} \in \mathbb{R}^{d\times n}$ is any matrix that allows rotation. The column vectors are of

equal Euclidean norm and any pair has an inner product of $\boldsymbol{e}_i^\top \cdot \boldsymbol{e}_j = -\frac{1}{N-1}$ for $i \neq j$, which together ensures pairwise maximum distances. Finally, we assign the features $\boldsymbol{h}_j$ to classes by determining the respective closest centroid from $\boldsymbol{E}$, which leads to a partitioning $\boldsymbol{H}_1, ..., \boldsymbol{H}_N$ of $\boldsymbol{H}$. An illustration of this strategy is provided in Figure 2 a).

Task $\mathcal{T}^i$. At the start of each task, we need to introduce new neural meshes. Following Wang *et al.* [52], for each new class c we initialize $\mathfrak{N}_c$ as a cuboid where its dimensions are determined from ground-truth meshes and vertices are sampled on a regular grid on the surface. As illustrated in Figure 2 b), we then pick the partition $\boldsymbol{H}_c$ of initial features and randomly assign them to the vertices of the new mesh Θ_c. We initialize the feature extractor Φ_0 with unsupervised pre-training using DINO-v1 [5]. As shown in Figure 1, to train for a new task, we make a copy $\Phi_i = \Phi_{i-1}$ and then leverage Φ_{i-1} for knowledge distillation. If available, we discard any old network Φ_{i-2}.

4.2 Optimization

Positional Regularization. To ensure that our latent space maintains the initial partitioning over time, we introduce a penalty of the distance of the neural features Θ_c to their corresponding class centroid $\boldsymbol{e}_c$:

$$\mathcal{L}_{\text{etf}} = -\sum_k o_c^k \cdot \log\left(\frac{e^{\kappa_2(f_{\pi_c(k)}^\top \cdot \boldsymbol{e}_c)}}{\sum_{\boldsymbol{e}_m \in \boldsymbol{E}} e^{\kappa_2(f_{\pi_c(k)}^\top \cdot \boldsymbol{e}_m)}}\right). \tag{7}$$

This is illustrated in Figure 2 c).

Continual Training Loss. We denote any unused partitions in $\boldsymbol{H}$ with $\bar{\boldsymbol{H}}$ and limit the spread of the neural meshes in the current task to refrain from $\bar{\boldsymbol{H}}$ by posing the following additional contrastive loss:

$$\mathcal{L}_{\text{cont}} = -\sum_k o_c^k \cdot \log\left(\frac{e^{\kappa_1(f_{\pi_c(k)}^\top \cdot \theta_c^k)}}{\sum_{\theta^m \in \bar{\theta}_k} e^{\kappa_1(f_{\pi_c(k)}^\top \cdot \theta^m)} + \sum_{h_j \in \bar{\boldsymbol{H}}} e^{\kappa_1(f_{\pi_c(k)}^\top \cdot h_j)}}\right). \tag{8}$$

The denominator is split into two parts, where the first one minimizes Equation 3 and the second part corresponds to the additional constraint imposed by the features in the unused partitions $\bar{\boldsymbol{H}}$. This is illustrated in Figure 2 d).

Knowledge Distillation. To mitigate forgetting, as indicated in Figure 1, we additionally use a distillation loss after the initial task. The new inputs are also fed through the frozen backbone Φ_{i-1} of the previous task to obtain its feature map. Specifically, let $\hat{f}_{\pi_c(k)}$ denote the old feature for the vertex k. To

distill classes from previous tasks into Φ_i, we formulate the distillation using the Kullback-Leibler divergence:

$$\mathcal{L}_{\text{kd}} = -\sum_{k}\sum_{m} p_m(\hat{f}_{\pi_c(k)}) \log(\frac{p_m(\hat{f}_{\pi_c(k)})}{p_m(f_{\pi_c(k)})}), \tag{9}$$

where:

$$p_m(f_{\pi_c(k)}) = \frac{e^{\kappa_3(f_{\pi_c(k)}^\top \cdot \theta_m)}}{\sum_{\theta_m \in \Theta^{i-1}} e^{\kappa_3(f_{\pi_c(k)}^\top \cdot \theta_m)}}. \tag{10}$$

Note that, unless we are considering an exemplar of a previous task, the real corresponding feature θ_c^k is not even considered in this formulation. However, the aim here is not to optimize Φ_i for the current task, but to extract the dark knowledge [16] from Φ_{i-1} about classes from previous tasks. Consequently, the concentration $\kappa_3 < 1$ has to be small to get usable gradients from all likelihoods.

Continual Training. During training of Φ_i, we optimize the combined training objective:

$$\mathcal{L} = \mathcal{L}_{\text{cont}} + \lambda_{\text{etf}}\mathcal{L}_{\text{etf}} + \lambda_{\text{kd}}\mathcal{L}_{\text{kd}}, \tag{11}$$

where λ_{etf} and λ_{kd} are weighting parameters.

4.3 Exemplar Selection

At each training stage, we randomly remove exemplars from old classes to equally divide our replay buffer for the current number of classes. Xiang *et al.* [61] showed that certain classes are heavily biased towards certain viewing angles. Therefore, to increase the robustness and accuracy for rarely appearing view directions, we propose an exemplar selection strategy that takes viewing angles into account. Assuming we want to integrate a new class and the available slots for it are m, we build a b-bin histogram across the azimuth angles and randomly select $\lfloor m/b \rfloor$ exemplars for each bin. When insufficient exemplars are available for a bin we merge it together with a neighboring one. In case the process yields less than m exemplars in total, we fill up remaining slots with random samples. When reducing the exemplar sets, we evenly remove samples from each bin to maintain the balance across the azimuth angle distribution.

4.4 Inference

Classification. Following Jesslen *et al.* [19], we perform classification via a vertex matching approach. For each feature f_i in the produced feature map of Φ, we compute its similarities to the foreground (Θ) and background ($\mathcal{B}$) models. We define the background score s_β^i and the class scores s^i for each class c as

$$s_c^i = \max_{\theta_c^l \in \Theta_c} f_i^\top \cdot \theta_c^l, \tag{12}$$

$$s_\beta^i = \max_{\beta^l \in \mathcal{B}} f_i^\top \cdot \beta^l, \tag{13}$$

where we identify a feature as being in the foreground $\mathcal{F}$, if there is at least one $s_c^i > s_\beta^i$ and classify based on the foreground pixels only.

In contrast to Jesslen *et al.* [19], we additionally include an uncertainty term to reduce the influence of features that can not be identified with high confidence. In the following, we denote the $n-$th largest class score for feature f_i as $max_{S^i}^{(n)}$. The final score of class y is then given as

$$s_y = \max_{i \in \mathcal{F}} \left[s_y^i - (1 - (max_{s^i}^{(1)} - max_{s^i}^{(2)})) \right], \tag{14}$$

where the subtracted term indicates a measure of confusion estimated based on the difference of the two highest class scores for foreground feature f_i. The predicted category is then simply the class c that maximizes this score.

Pose Estimation. For pose estimation we use the same render-and-compare approach as Wang *et al.* [52] together with the template matching proposed by Jesslen *et al.* [19] for speedup. For more information about the pose estimation, we refer the reader to the supplemental material.

5 Experiments

In the following, we explain the experimental setup and then discuss the results of our incremental neural mesh models for image classification and 3D pose estimation on both, in-domain and OOD datasets. For a comprehensive ablation study of all components of our model, we refer to the supplemental material.

5.1 Datasets and Implementation Details

In-Domain-Datasets. PASCAL3D+ [62] (P3D) has high-quality camera pose annotations with mostly unoccluded objects, making it ideal for our setting. However, with only 12 classes it is small compared to other datasets used in continual learning [24, 46]. **ObjectNet3D** [60] (O3D) contains 100 classes and presents a significantly more difficult setting. Camera pose annotations are less reliable and the displayed objects can be heavily occluded or truncated, making both the vertex mapping and the update process noisy.

OOD-Datasets. The **Occluded-PASCAL3D+** [55] (O-P3D) and **corrupted-PASCAL3D+** (C-P3D) datasets are variations of original P3D and consist of a test dataset only. In the O-P3D dataset, parts of the original test datasets have been artificially occluded by superimposing occluders on the images with three different levels: L1 ($20\% - 40\%$), L2 ($40\% - 60\%$) and L3 ($60\% - 80\%$). The C-P3D dataset, on the other hand, follows [14] and tests robustness against image corruptions. We evaluate 19 different corruptions with a severity of 4 out of 5, using the `imagecorruptions` [38] library. Finally, we consider the **OOD-CV** [64] dataset, which provides a multitude of severe domain shifts.

Table 1: Average classification accuracies on Pascal3D (P3D) and ObjectNet3D (O3D). Training data has been split into a base task (denoted Bn for size n) and evenly sized increments (denoted $+n$ for size n). As visible, our method consistently outperforms the baselines by a significant margin.

Metric	Method	Repr.	*P3D*		*O3D*		
			$B0+6$	$B0+3$	$B0+20$	$B0+10$	$B50+10$
Classification $acc(1:i)$ in % ↑	LWF	R50.	93.83	89.34	67.78	48.82	46.73
	FeTrIL	R50.	95.64	96.82	67.18	70.34	70.43
	FeCAM	R50.	84.85	64.36	67.96	69.59	72.21
	iCaRL	R50	97.1	93.80	72.55	57.46	64.02
	DER	R50	96.69	94.18	78.55	76.33	75.17
	Podnet	R50	95.13	91.71	71.96	65.21	72.98
	Ours	NeMo	**98.82**	**98.21**	**89.25**	**88.85**	**84.20**

Implementational Details. We choose a ResNet50 architecture for our feature extractor Φ with two upsampling layers and skip connections, resulting in a final feature map at $\frac{1}{8}$ of the input resolution. Each neural mesh $\mathfrak{N}_y$ contains approximately $1,100$ uniformly distributed vertices with a neural texture of dimension $d = 128$. We train for 50 epochs per task, with a learning rate of $1e-5$ that is halved after 10 epochs. The replay buffer can store up to 240 and $2,000$ samples for P3D and O3D respectively. Our feature extractor is optimized using Adam with default parameters and the neural textures Θ are updated with momentum of $\eta = 0.9$. During pose estimation, we initialize the camera pose using template matching as proposed by [19] and optimize it with PyTorch3D's differentiable rasterizer [44]. The initial camera pose is refined by minimizing the reconstruction loss between the feature map produced by Φ and the rendered mesh. We use Adam with a learning rate of 0.05 for 30 total epochs and a distance threshold $R = 48$ to measure the neighborhood $\mathcal{N}_c^k$ in Equation 8. Each term in the combined loss in Equation 11 is assigned a weighting and concentration parameter. The weighting parameters are $\lambda_{\text{etf}} = 0.2$ and $\lambda_{\text{kd}} = 2.0$ and as concentration parameters we choose $\kappa_1 = 1/0.07 \approx 14.3$, $\kappa_2 = 1$, and $\kappa_3 = 0.5$. We provide further details on the training settings of the baselines in the supplemental material.

Evaluation. We evaluate our method and its baselines on both, class-incremental classification and class-incremental pose estimation. For the methods trained on P3D, we evaluate on the P3D test dataset, the O-P3D dataset, and the C-P3D dataset. When training on O3D or OOD-CV, we evaluate on their corresponding test dataset only. For classification, we follow previous work [26, 31, 45] and consider the mean accuracy over all tasks $acc(1:i)$ of Φ_i, after training on $\mathcal{T}^i$ on test dataset $\mathcal{D}$ for classes $1..i$. The 3D pose of an object can be represented with azimuth, elevation, and roll angle. We measure the deviation of predicted and ground-truth pose in terms of these angles according to the error of the predicted

Table 2: Average classification and pose estimation accuracies on Pascal3D (P3D) and its variants. As visible, iNeMo outperforms all 2D baselines consistently for classification and by an especially large margin for the OOD and strong occlusion cases. We also present the first approach for incremental pose estimation and outperform other methods in most cases for $\pi/6$, while we consistently outperform them for the tighter error bound $\pi/18$. Note that for all evaluations except OOD-CV, we use the model trained on 4 tasks that is also displayed in Figure 3. As OOD-CV provides a separate training set of 10 classes, we consider 2 tasks with 5 classes.

Metric	Method	Repr.	*P3D*	*Occluded P3D*			*C-P3D*	*OOD-CV*
				*L*1	*L*2	*L*3		
	LWF	R50	89.34	30.58	21.64	14.65	66.17	57.61
	FeTrIL	R50	96.82	88.34	76.28	55.89	39.02	63.74
$acc(1:i)$ in % ↑	FeCAM	R50	84.85	52.53	42.75	34.28	42.38	56.05
Classification	iCaRL	R50	93.80	34.95	26.00	16.93	76.24	61.80
	DER	R50	94.18	49.70	36.86	22.76	69.56	56.35
	PODNet	R50	91.91	42.40	32.99	22.43	68.46	57.10
	Ours	NeMo	**98.21**	**94.19**	**87.20**	**71.55**	**83.09**	**80.82**
	LwF	R50	53.47	44.58	39.77	**36.61**	53.55	30.65
Pose $\pi/6$	iCaRL	R50	57.74	44.03	38.15	33.52	**54.57**	28.71
$acc(1:i)$ in % ↑	Ours	NeMo	**79.28**	**64.71**	**52.26**	34.01	47.30	**33.75**
	LwF	R50	20.33	12.03	8.38	5.52	17.29	8.04
Pose $\pi/18$	iCaRL	R50	22.76	11.04	7.33	4.56	17.81	8.04
$acc(1:i)$ in % ↑	Ours	NeMo	**51.73**	**35.53**	**26.88**	**10.672**	**23.02**	**12.8**

and the ground-truth rotation matrix $\Delta(R_{\text{pred}}, R_{\text{gt}}) = \|\log m(R_{\text{pred}}^{\top} R_{\text{gt}})\|_F / \sqrt{2}$ as proposed by [66]. Following previous work [52, 66], we report the accuracy according to the thresholds $\pi/6$ and $\pi/18$.

Baselines. For the task of class-incremental learning, we compare against a collection of replay-based and replay-free methods. For the replay-based methods, we choose the seminal work iCaRL [45], and the more recent PODNet [11] and DER [63]. For replay-free methods, we choose the seminal work LwF [26] and the two state-of-the-art methods FeTrIL [41] and FeCAM [12]. All approaches are implemented using the PyCIL library [65] and trained with the original hyperparameters as in [65]. For a fair comparison of all methods, we use the ResNet-50 backbone initialized with DINO-v1 [5] pre-trained weights.

To the best of our knowledge, incremental pose estimation with a class-agnostic backbone has not been explored before. We define incremental pose estimation baselines by discretizing the polar camera coordinates and formulate pose estimation as a classification problem following [66]. More specifically, we define 42 bins for each azimuth, elevation and roll angle, making it a $42 \cdot 3 = 126$ class classification problem [66]. This allows a straightforward extension of conventional class-incremental learning techniques to the setting of pose estimation.

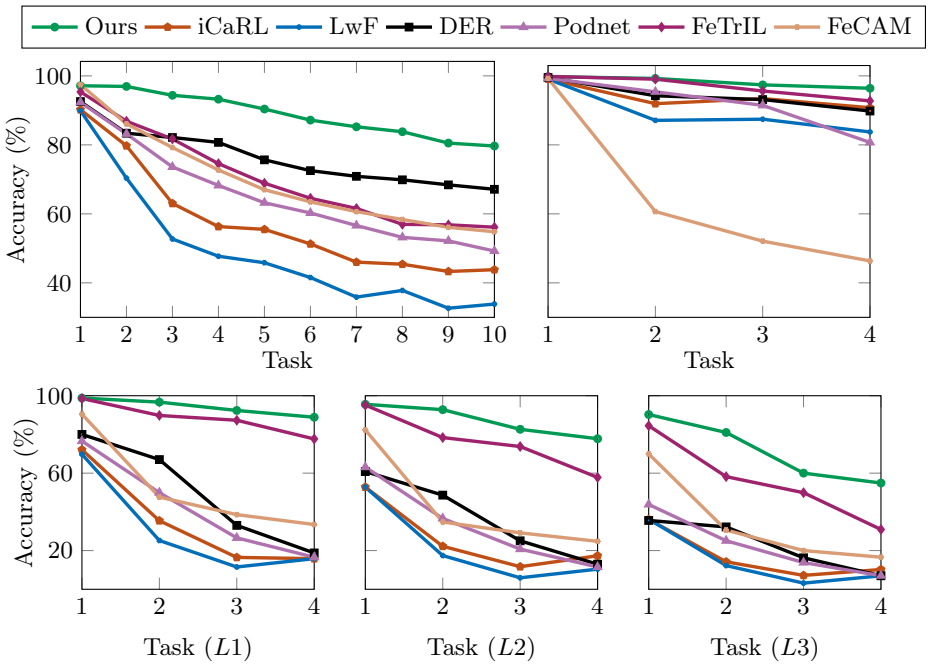

Fig. 3: Comparison of classification performance decay over tasks for our method and the baselines. **Top-Left:** Results for O3D (100 classes) split into 10 even tasks. **Top-Right:** Results for P3D (12 classes) split into 4 even tasks. **Bottom:** Results for O-P3D with occlusion levels L1, L2 and L3 after each task. One can observe that our method outperforms all other methods. Especially in the occluded cases, our method outperforms them by a very large margin up to 70%, even still showing strong performance for the largest occlusion level L3 with $60-80\%$ occlusions.

We provide such class-incremental pose estimation results using the training procedure of iCaRL [45] and LwF [26]. Both methods are trained for 100 epochs per task using SGD with a learning rate of 0.01 as proposed by [66].

5.2 Robust Class-Incremental Classification

In Table 1, we provide the in-distribution classification results for P3D and O3D. Our method outperforms the baselines in all cases, including the harder O3D setting with 100 classes. Furthermore, Table 2 shows the comparison of class-incremental results on all P3D variants for both, classification and 3D pose estimation. As visible, our method outperforms the other methods with a large margin under domain shifts: for the L3 occluded case, it is larger than 48%, for the corrupted C-P3D it is larger than 6%, and for the OOD-CV dataset it is larger than 19%. Figure 3 shows task-wise accuracy on the O3D/P3D dataset for 10 and 4 even tasks respectively, as well as the task-wise accuracy on the O-P3D

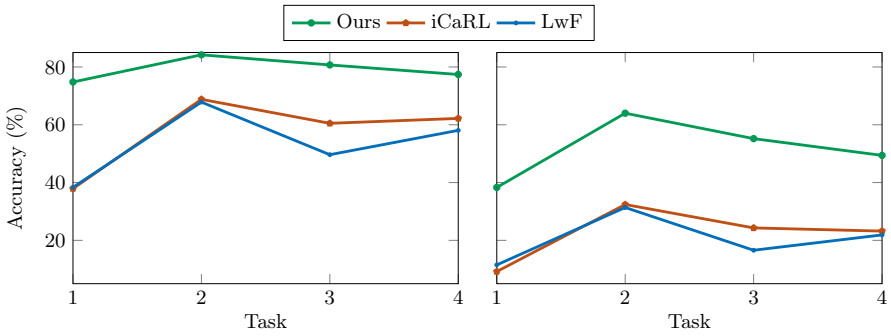

Fig. 4: Comparison of the task-wise pose estimation accuracy on P3D for 4 even tasks, where we show the thresholds **left:** $\pi/6$ and **right:** $\pi/18$. One can observe that our method outperforms all other methods and retains high pose estimation accuracy throughout the incremental training process. One can also observe that for pose estimation, there is a stronger dependence on the difficulty of the considered classes instead of the method's ability to retain knowledge.

dataset for all occlusion levels. The same observation as before can be made, where our method exhibits significantly less performance decay over new tasks. This overall demonstrates that our incremental neural mesh models outperform their baselines decisively in robustness.

5.3 Class Incremental Pose Estimation

Table 2 also shows that our method significantly outperforms both ResNet-50 based methods for the task of incremental pose estimation. As visible, the feature representation learned by the 2D pose estimation networks is much less affected by both, catastrophic forgetting and domain shifts. Figure 4 shows that the performance decrease is much less severe across all tasks, where the difference in performance is much more dependent on the difficulty of the considered classes instead of the method's ability to retain knowledge.

6 Conclusions

In this work, we introduce incremental neural mesh models, which enable robust class-incremental learning for both, image classification and 3D pose estimation. For the first time, we present a model that can learn new prototypical 3D representations of object categories over time. The extensive evaluation on Pascal3D and ObjectNet3D shows that our approach outperforms all baselines even in the in-domain setting and surpasses them by a large margin in the OOD case. We also introduced the first approach for class-incremental learning of pose estimation. The results overall demonstrate the fundamental advantage of 3D object-centric representations, and we hope that this will spur a new line of research in the community.

Acknowledgements

We gratefully acknowledge the stimulating research environment of the GRK 2853/1 “Neuroexplicit Models of Language, Vision, and Action”, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project number 471607914. Adam Kortylewski gratefully acknowledges support for his Emmy Noether Research Group, funded by the German Research Foundation (DFG) under Grant No. 468670075. Alan L. Yuille gratefully acknowledges the Army Research Laboratory award W911NF2320008 and ONR N00014-21-1-2812.

References

1. Aljundi, R., Chakravarty, P., Tuytelaars, T.: Expert gate: Lifelong learning with a network of experts. In: CVPR. pp. 3366–3375 (2017)
2. Aljundi, R., Kelchtermans, K., Tuytelaars, T.: Task-free continual learning. In: CVPR. pp. 11254–11263 (2019)
3. Bai, Y., Wang, A., Kortylewski, A., Yuille, A.: Coke: Localized contrastive learning for robust keypoint detection. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (2023)
4. Bang, J., Kim, H., Yoo, Y., Ha, J.W., Choi, J.: Rainbow memory: Continual learning with a memory of diverse samples. In: CVPR. pp. 8218–8227 (2021)
5. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the International Conference on Computer Vision (ICCV) (2021)
6. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: ECCV. pp. 233–248 (2018)
7. Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: ECCV. pp. 532–547 (2018)
8. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with A-GEM. In: ICLR (2019)
9. Chen, Z., Liu, B.: Lifelong machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **12**(3), 1–207 (2018)
10. De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. TPAMI **44**(7), 3366–3385 (2021)
11. Douillard, A., Cord, M., Ollion, C., Robert, T., Valle, E.: Podnet: Pooled outputs distillation for small-tasks incremental learning. In: ECCV. pp. 86–102 (2020)
12. Goswami, D., Liu, Y., Twardowski, B., van de Weijer, J.: Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. In: Advances in Neural Information Processing Systems. vol. 36 (2024)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
14. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: ICLR (2019)
15. Hendrycks, D., Mu, N., Cubuk, E.D., Zoph, B., Gilmer, J., Lakshminarayanan, B.: Augmix: A simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781 (2019)

16. Hinton, G., Vinyals, O., Dean, J., et al.: Distilling the knowledge in a neural network. In: NIPS Workshops (2014)
17. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: CVPR. pp. 831–839 (2019)
18. Iwase, S., Liu, X., Khirodkar, R., Yokota, R., Kitani, K.M.: Repose: Fast 6d object pose refinement via deep texture rendering. In: ICCV. pp. 3303–3312 (2021)
19. Jesslen, A., Zhang, G., Wang, A., Yuille, A., Kortylewski, A.: Robust 3d-aware object classification via discriminative render-and-compare. arXiv preprint arXiv:2305.14668 (2023)
20. Joseph, K.J., Khan, S., Khan, F.S., Anwer, R.M., Balasubramanian, V.N.: Energy-based latent aligner for incremental learning. In: CVPR. pp. 7452–7461 (2022)
21. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. PNAS pp. 3521–3526 (2017)
22. Kortylewski, A., Liu, Q., Wang, A., Sun, Y., Yuille, A.: Compositional convolutional neural networks: A robust and interpretable model for object recognition under occlusion. IJCV pp. 1–25 (2020)
23. Kouros, G., Shrivastava, S., Picron, C., Nagesh, S., Chakravarty, P., Tuytelaars, T.: Category-level pose retrieval with contrastive features learnt with occlusion augmentation. arXiv preprint arXiv:2208.06195 (2022)
24. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Technical Report TR-2009 (2009)
25. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. In: ECCV. pp. 683–698 (2018)
26. Li, Z., Hoiem, D.: Learning without forgetting. TPAMI **40**(12), 2935–2947 (2018)
27. Liu, Y., Li, Y., Schiele, B., Sun, Q.: Online hyperparameter optimization for class-incremental learning. In: AAAI (2023)
28. Liu, Y., Li, Y., Schiele, B., Sun, Q.: Wakening past concepts without past data: Class-incremental learning from online placebos. In: WACV. pp. 2226–2235 (2024)
29. Liu, Y., Schiele, B., Sun, Q.: Adaptive aggregation networks for class-incremental learning. In: CVPR. pp. 2544–2553 (2021)
30. Liu, Y., Schiele, B., Sun, Q.: RMM: reinforced memory management for class-incremental learning. In: NeurIPS. pp. 3478–3490 (2021)
31. Liu, Y., Su, Y., Liu, A., Schiele, B., Sun, Q.: Mnemonics training: Multi-class incremental learning without forgetting. In: CVPR. pp. 12245–12254 (2020)
32. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV. pp. 10012–10022 (2021)
33. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. In: NIPS. pp. 6467–6476 (2017)
34. Luo, Z., Liu, Y., Schiele, B., Sun, Q.: Class-incremental exemplar compression for class-incremental learning. In: CVPR. pp. 11371–11380 (2023)
35. Ma, W., Wang, A., Yuille, A.L., Kortylewski, A.: Robust category-level 6d pose estimation with coarse-to-fine rendering of neural features. In: ECCV. pp. 492–508 (2022)
36. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of Learning and Motivation, vol. 24, pp. 109–165. Elsevier (1989)
37. McRae, K., Hetherington, P.: Catastrophic interference is eliminated in pre-trained networks. In: CogSci (1993)

38. Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., Bringmann, O., Ecker, A.S., Bethge, M., Brendel, W.: Benchmarking robustness in object detection: Autonomous driving when winter is coming. arXiv preprint arXiv:1907.07484 (2019)
39. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: CVPR. pp. 7074–7082 (2017)
40. Papyan, V., Han, X., Donoho, D.L.: Prevalence of neural collapse during the terminal phase of deep learning training. Proceedings of the National Academy of Sciences **117**(40), 24652–24663 (2020)
41. Petit, G., Popescu, A., Schindler, H., Picard, D., Delezoide, B.: Fetril: Feature translation for exemplar-free class-incremental learning. In: CVPR (2023)
42. PourKeshavarzi, M., Zhao, G., Sabokrou, M.: Looking back on learned experiences for class/task incremental learning. In: ICLR (2022)
43. Prabhu, A., Torr, P.H., Dokania, P.K.: GDumb: A simple approach that questions our progress in continual learning. In: ECCV. pp. 524–540 (2020)
44. Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.Y., Johnson, J., Gkioxari, G.: Accelerating 3d deep learning with pytorch3d. arXiv:2007.08501 (2020)
45. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: iCaRL: Incremental classifier and representation learning. In: CVPR. pp. 5533–5542 (2017)
46. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**, 211–252 (2015)
47. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: NeurIPS. pp. 2990–2999 (2017)
48. Simon, C., Koniusz, P., Harandi, M.: On learning the geodesic path for incremental learning. In: CVPR. pp. 1591–1600 (2021)
49. Tao, X., Chang, X., Hong, X., Wei, X., Gong, Y.: Topology-preserving class-incremental learning. In: ECCV. pp. 254–270 (2020)
50. Tulsiani, S., Malik, J.: Viewpoints and keypoints. In: CVPR (June 2015)
51. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NeurIPS **30** (2017)
52. Wang, A., Kortylewski, A., Yuille, A.: NeMo: Neural mesh models of contrastive features for robust 3d pose estimation. ICLR (2021)
53. Wang, A., Ma, W., Yuille, A., Kortylewski, A.: Neural textured deformable meshes for robust analysis-by-synthesis. In: WACV. pp. 3108–3117 (2024)
54. Wang, A., Mei, S., Yuille, A.L., Kortylewski, A.: Neural view synthesis and matching for semi-supervised few-shot learning of 3d pose. NeurIPS **34**, 7207–7219 (2021)
55. Wang, A., Sun, Y., Kortylewski, A., Yuille, A.L.: Robust object detection under occlusion with context-aware compositionalnets. In: CVPR. pp. 12645–12654 (2020)
56. Wang, A., Wang, P., Sun, J., Kortylewski, A., Yuille, A.: Voge: a differentiable volume renderer using gaussian ellipsoids for analysis-by-synthesis. In: ICLR (2022)
57. Wang, F.Y., Zhou, D.W., Ye, H.J., Zhan, D.C.: Foster: Feature boosting and compression for class-incremental learning. In: ECCV (2022)
58. Wu, C., Herranz, L., Liu, X., Van De Weijer, J., Raducanu, B., et al.: Memory replay gans: Learning to generate new categories without forgetting. NeurIPS **31** (2018)
59. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: CVPR. pp. 374–382 (2019)
60. Xiang, Y., Kim, W., Chen, W., Ji, J., Choy, C., Su, H., Mottaghi, R., Guibas, L., Savarese, S.: Objectnet3d: A large scale database for 3d object recognition. In: ECCV (2016)

61. Xiang, Y., Mottaghi, R., Savarese, S.: Beyond pascal: A benchmark for 3d object detection in the wild. In: WACV. pp. 75–82. IEEE (2014)
62. Xiang, Y., Mottaghi, R., Savarese, S.: Beyond pascal: A benchmark for 3d object detection in the wild. In: WACV (2014)
63. Yan, S., Xie, J., He, X.: Der: Dynamically expandable representation for class incremental learning. In: CVPR. pp. 3014–3023 (2021)
64. Zhao, B., Yu, S., Ma, W., Yu, M., Mei, S., Wang, A., He, J., Yuille, A., Kortylewski, A.: Ood-cv: A benchmark for robustness to individual nuisances in real-world out-of-distribution shifts. In: ECCV (2022)
65. Zhou, D., Wang, F., Ye, H., Zhan, D.: Pycil: a python toolbox for class-incremental learning. Sci. China Inf. Sci. **66**(9) (2023)
66. Zhou, X., Karpur, A., Luo, L., Huang, Q.: Starmap for category-agnostic keypoint and viewpoint estimation. In: ECCV. pp. 318–334 (2018)