# FEW-SHOT LEARNING VIA TENSOR HALLUCINATION

**Michalis Lazarou**[1]  **Yannis Avrithis**[2]  **Tania Stathaki**[1]
[1]Imperial College London
[2]Inria, Univ Rennes, CNRS, IRISA

## ABSTRACT

Few-shot classification addresses the challenge of classifying examples given only limited labeled data. A powerful approach is to go beyond data augmentation, towards data synthesis. However, most of data augmentation/synthesis methods for few-shot classification are overly complex and sophisticated, *e.g.* training a wGAN with multiple regularizers or training a network to transfer latent diversities from known to novel classes. We make two contributions, namely we show that: (1) using a simple loss function is more than enough for training a feature generator in the few-shot setting; and (2) learning to generate tensor features instead of vector features is superior. Extensive experiments on *mini*Imagenet, CUB and CIFAR-FS datasets show that our method sets a new state of the art, outperforming more sophisticated few-shot data augmentation methods.

## 1 INTRODUCTION

Deep learning continuously keeps improving the state of the art in multiple different fields, such as natural language understanding Mikolov et al. (2010) and computer vision Krizhevsky et al. (2012). However, even though the success of deep learning models is undeniable, a fundamental limitation is their dependence on large amounts of labeled data. This limitation inhibits the application of state of the art deep learning methods to real-world problems, where the cost of annotating data is high and data can be scarce, *e.g.* rare species classification.

To address this limitation, *few-shot learning* has attracted significant interest in recent years. One of the most common lines of research is *meta-learning*, where training episodes mimic a few-shot task by having a small number of classes and a limited number of examples per class. Meta-learning approaches can further be partitioned in *optimization-based*, learning to update the learner's meta-parameters Finn et al. (2017); Zintgraf et al.; Nichol et al. (2018); Ravi & Larochelle (2016), *metric-based*, learning a discriminative embedding space where novel examples are easy to classify Koch et al. (2015); Snell et al. (2017); Sung et al. (2018); Vinyals et al. (2016) and *model-based*, depending on specific model architectures to learn how to update the learner's parameters effectively Santoro et al. (2016); Munkhdalai & Yu (2017).

Beyond meta-learning, other approaches include leveraging the *manifold structure* of the data, by label propagation, embedding propagation or graph neural networks Garcia & Bruna (2017); Kim et al. (2019); Liu et al. (2018); Lazarou et al. (2020); and *domain adaptation*, reducing the domain shift between source and target domains Dong & Xing (2018); Hsu et al. (2017). Another line of research is *data augmentation*, addressing data deficiency by augmenting the few-shot training dataset with extra examples in the image space Chen et al. (2019); Zhang et al. (2018) and in the feature space Chen et al. (2019); Li et al. (2020); Luo et al. (2021). Such methods go beyond standard augmentation Krizhevsky et al. (2012) towards *synthetic data generation* and *hallucination*, achieving a greater extent of diversity.

Our work falls into the category of data augmentation in the feature space. We show that using a simple loss function to train a feature hallucinator can outperform other state of the art few-shot data augmentation methods that use more complex and sophisticated generation methods, such as wGAN Li et al. (2020), VAE Luo et al. (2021) and networks trained to transfer example diversity Chen et al. (2020). Also, to the best of our knowledge, we are the first to propose generating *tensor features* instead of vector features in the few-shot setting.

## 2 METHOD

### 2.1 PROBLEM FORMULATION

We are given a labeled dataset $D_{\text{base}} := \{(x_i, y_i)\}_{i=1}^{I}$, with each example $x_i$ having a label $y_i$ in one of the classes in $C_{\text{base}}$. This dataset is used to learn a parametrized mapping $f_\theta : \mathcal{X} \to \mathbb{R}^{d \times h \times w}$ from an input image space $\mathcal{X}$ to a *feature* or *embedding* space, where *feature tensors* have $d$ dimensions (channels) and spatial resolution $h \times w$ (height $\times$ width).

The knowledge acquired at representation learning is used to solve *novel tasks*, assuming access to a dataset $D_{\text{novel}}$, with each example being associated with one of the classes in $C_{\text{novel}}$, where $C_{\text{novel}}$ is disjoint from $C_{\text{base}}$. In *few-shot classification* Vinyals et al. (2016), a novel task is defined by sampling a *support set* $S$ from $D_{\text{novel}}$, consisting of $N$ classes with $K$ labeled examples per class, for a total of $L := NK$ examples. Given the mapping $f_\theta$ and the support set $S$, the problem is to learn an $N$-way classifier that makes predictions on unlabeled *queries*, also sampled from $D_{\text{novel}}$. Queries are treated independently of each other. This is referred to as *inductive inference*.

### 2.2 REPRESENTATION LEARNING

The goal of *representation learning* is to learn the embedding function $f_\theta$ that can be applied to $D_{\text{novel}}$ to extract embeddings and solve novel tasks. We use $f_\theta$ followed by *global average pooling* (GAP) and a parametric *base classifier* $c_\phi$ to learn the representation. We denote by $\bar{f}_\theta : \mathcal{X} \to \mathbb{R}^d$ the composition of $f_\theta$ and GAP. We follow the two-stage regime by Tian et al. (2020) to train our embedding model. In the first stage, we train $f_\theta$ on $D_{\text{base}}$ using standard cross-entropy loss $L_{\text{CE}}$:

$$L(D_{\text{base}}; \theta, \phi) := \sum_{i=1}^{I} L_{\text{CE}}(c_\phi(\bar{f}_\theta(x_i)), y_i) + R(\phi), \tag{1}$$

where $R$ is a regularization term. In the second stage, we adopt a *self-distillation* process: The embedding model $f_\theta$ and classifier $c_\phi$ from the first stage serve as the teacher and we distill their knowledge to a new student model $f_{\theta'}$ and classifier $c_{\phi'}$, with identical architecture. The student is trained using a linear combination of the standard cross-entropy loss, as in stage one, and the Kullback-Leibler (KL) divergence between the student and teacher predictions:

$$L_{\text{KD}}(D_{\text{base}}; \theta', \phi') := \alpha L(D_{\text{base}}; \theta', \phi') + \beta \text{KL}(c_{\phi'}(\bar{f}_{\theta'}(x_i)), c_\phi(\bar{f}_\theta(x_i))), \tag{2}$$

where $\alpha$ and $\beta$ are scalar weights and $\theta, \phi$ are fixed.

### 2.3 FEATURE TENSOR HALLUCINATOR

All existing feature hallucination methods are trained using vector features, losing significant spatial and structural information. By contrast, our hallucinator is trained on the tensor features before global average pooling and generates tensor features as well. In particular, we use the student model $f_{\theta'} : \mathcal{X} \to \mathbb{R}^{d \times h \times w}$, pre-trained using (2), as our embedding network to train our tensor feature hallucinator. The hallucinator consists of two networks: a *conditioner* network $h$ and a *generator* network $g$. The conditioner aids the generator in generating class-conditional examples. Given a set $\{x_i^j\}_{i=1}^{K}$ of examples associated with class $j$ for $j = 1, \dots, N$, conditioning is based on the *prototype tensor* $p_j \in \mathbb{R}^{d \times h \times w}$ of each class $j$,

$$p_j := \frac{1}{K} \sum_{i=1}^{K} f_{\theta'}(x_i^j). \tag{3}$$

The conditioner $h : \mathbb{R}^{d \times h \times w} \to \mathbb{R}^{d'}$ maps the prototype tensor to the *class-conditional vector* $s_j := h(p_j) \in \mathbb{R}^{d'}$. The generator $g : \mathbb{R}^{k+d'} \to \mathbb{R}^{d \times h \times w}$ takes as input this vector as well as a $k$-dimensional sample $z \sim \mathcal{N}(\mathbf{0}, I_k)$ from a standard normal distribution and generates a *class-conditional tensor feature* $g(z; s_j) \in \mathbb{R}^{d \times h \times w}$ for class $j$.

### 2.4 TRAINING THE HALLUCINATOR

We train our hallucinator using a meta-training regime, similar to Li et al. (2020); Chen et al. (2019); Schwartz et al. (2018). At every iteration, we sample a new episode by randomly sampling $N$ classes

and $K$ examples $X_j := \{x_i^j\}_{i=1}^K$ for each class $j$ from $D_{\text{base}}$. We obtain the prototype tensor $p_j$ for each class $j$ by (3) and the class-conditional vector $s_j := h(p_j)$ by the conditioner $h$. For each class $j$, we draw $M$ samples $\{z_m\}_{m=1}^M$ from a normal distribution $\mathcal{N}(\mathbf{0}, I_k)$ and we generate $M$ class-conditional tensor features $\{g(z_m; s_j)\}_{m=1}^M$ using the generator $g$. We train our hallucinator $\{h, g\}$ on the episode data $X := \{X_j\}_{j=1}^N$ by minimizing the *mean squared error* (MSE) of generated class-conditional tensor features of class $j$ to the corresponding class prototype $p_j$:

$$L_{\text{hal}}(X; h, g) = \frac{1}{MN} \sum_{j=1}^N \sum_{m=1}^M \|g(z_m; h(p_j)) - p_j\|^2. \tag{4}$$

## 2.5 INFERENCE

At inference, we are given a few-shot task with a support set $S$ of $N$ classes with $K$ examples $S_j := \{x_i^j\}_{i=1}^K$ for each class $j$. We compute the tensor feature $f_{\theta'}(x_i^j) \in \mathbb{R}^{d \times h \times w}$ of each example using our trained backbone network $f_{\theta'}$ and obtain the prototype $p_j$ of each class $j$ by (3). Then, using our trained tensor feature hallucinator $\{h, g\}$, we generate $M$ class-conditional tensor features $G_j := \{g(z_m; h(p_j))\}_{m=1}^M$ for each class $j$, also in $\mathbb{R}^{d \times h \times w}$, where $z_m$ are drawn from $\mathcal{N}(\mathbf{0}, I_k)$. We augment the support features $f_{\theta'}(S_j)$ with the generated features $G_j$, resulting in $K + M$ labeled tensor features per class in total. We now apply GAP to those tensor features and obtain new, *vector class prototypes* in $\mathbb{R}^d$. Finally, we also apply GAP to the query tensor features and classify each query to the class of the nearest prototype.

## 3 EXPERIMENTS

### 3.1 SETUP

**Datasets**  We carry out experiments on three commonly used few-shot classication benchmark datasets: *mini*Imagenet, CUB and CIFAR-FS. Further details are provided in subsection A.1.

**Tasks**  We consider $N$-way, $K$-shot classification tasks with $N = 5$ randomly sampled novel classes and $K \in \{1, 5\}$ examples drawn at random per class as support set $S$, that is, $L = 5K$ examples in total. For the query set $Q$, we draw $15$ additional examples per class, that is, $75$ examples in total, which is the most common choice Liu et al. (2018); Li et al. (2019); Yu et al. (2020).

**Competitors**  We compare our method with state-of-the-art data augmentation methods for few-shot learning, including MetaGAN Zhang et al. (2018), $\Delta$-encoder Schwartz et al. (2018), salient network (SalNet) Zhang et al. (2019), diversity transfer network (DTN) Chen et al. (2020), dual TriNet Chen et al. (2019), image deformation meta-network (IDeMe-Net) Chen et al. (2019), adversarial feature hallucination network (AFHN) Li et al. (2020) and variational inference network (VI-Net) Luo et al. (2021).

**Networks**  Many recent competitors Chen et al. (2019); Chen et al. (2019); Li et al. (2020); Luo et al. (2021) use ResNet-18 as backbone embedding model. To perform as fair comparison as possible, we use the same backbone.

Our *tensor feature hallucinator* (TFH) consists of a conditioner network and a generator network. The *conditioner* $h : \mathbb{R}^{d \times h \times w} \to \mathbb{R}^{d'}$ consists of two convolutional layers with a ReLU activation in-between, followed by flattening and a fully-connected layer. The *generator* $g : \mathbb{R}^{k+d'} \to \mathbb{R}^{d \times h \times w}$ consists of concatenation of $z$ and $s_j$ into $(z; s_j) \in \mathbb{R}^{k+d'}$, followed by reshaping to $(k + d') \times 1 \times 1$ and three transpose-convolutional layers with ReLU activation functions in-between and a sigmoid function in the end. More details are provided in subsection A.2.

We also provide an improved solution, called TFH-ft, where our tensor feature hallucinator is *fine-tuned* on novel-class support examples at inference.

**Baselines**  To validate the benefit of generating tensor features, we also implement a *vector feature hallucinator* (VFH), where we use $\bar{f}_{\theta'} : \mathcal{X} \to \mathbb{R}^d$ including GAP (2) as embedding model. In this

| METHOD | BACKBONE | *mini*IMAGENET | | CUB | | CIFAR-FS | |
|---|---|---|---|---|---|---|---|
| | | 1-shot | 5-shot | 1-shot | 5-shot | 1-shot | 5-shot |
| MetaGAN Zhang et al. (2018) | ConvNet-4 | $52.71_{\pm 0.64}$ | $68.63_{\pm 0.67}$ | – | – | – | – |
| $\Delta$-Encoder† Schwartz et al. (2018) | VGG-16 | 59.90 | 69.70 | $69.80_{\pm 0.46}$ | $82.60_{\pm 0.35}$ | 66.70 | 79.80 |
| SalNet Zhang et al. (2019) | ResNet-101 | $62.22_{\pm 0.87}$ | $77.95_{\pm 0.65}$ | – | – | – | – |
| DTN Chen et al. (2020) | Resnet-12 | $63.45_{\pm 0.86}$ | $77.91_{\pm 0.62}$ | 72.00 | 85.10 | 71.50 | 82.80 |
| Dual TriNet Chen et al. (2019) | ResNet-18 | $58.80_{\pm 1.37}$ | $76.71_{\pm 0.69}$ | 69.61 | 84.10 | $63.41_{\pm 0.64}$ | $78.43_{\pm 0.64}$ |
| IDeMe-Net Chen et al. (2019) | ResNet-18 | $59.14_{\pm 0.86}$ | $74.63_{\pm 0.74}$ | – | – | – | – |
| AFHN Li et al. (2020) | ResNet-18 | $62.38_{\pm 0.72}$ | $78.16_{\pm 0.56}$ | $70.53_{\pm 1.01}$ | $83.95_{\pm 0.63}$ | $68.32_{\pm 0.93}$ | $81.45_{\pm 0.87}$ |
| VI-Net Luo et al. (2021) | ResNet-18 | 61.05 | 78.60 | 74.76 | 86.84 | – | – |
| Baseline (1) | ResNet-18 | $56.81_{\pm 0.81}$ | $78.31_{\pm 0.59}$ | $67.14_{\pm 0.89}$ | $86.22_{\pm 0.50}$ | $65.71_{\pm 0.95}$ | $84.68_{\pm 0.61}$ |
| Baseline-KD (2) | ResNet-18 | $59.62_{\pm 0.85}$ | $79.64_{\pm 0.62}$ | $70.85_{\pm 0.90}$ | $87.64_{\pm 0.48}$ | $69.15_{\pm 0.94}$ | $85.89_{\pm 0.59}$ |
| VFH (ours) | ResNet-18 | $61.92_{\pm 0.85}$ | $77.02_{\pm 0.64}$ | $75.25_{\pm 0.86}$ | $87.96_{\pm 0.48}$ | $72.60_{\pm 0.93}$ | $84.26_{\pm 0.67}$ |
| **TFH (ours)** | ResNet-18 | $\mathbf{64.25}_{\pm 0.85}$ | $80.10_{\pm 0.61}$ | $\mathbf{75.83}_{\pm 0.91}$ | $88.17_{\pm 0.48}$ | $73.88_{\pm 0.87}$ | $85.92_{\pm 0.61}$ |
| **TFH-ft (ours)** | ResNet-18 | $63.92_{\pm 0.86}$ | $\mathbf{80.41}_{\pm 0.60}$ | $75.39_{\pm 0.86}$ | $\mathbf{88.72}_{\pm 0.47}$ | $\mathbf{73.89}_{\pm 0.88}$ | $\mathbf{87.15}_{\pm 0.58}$ |

Table 1: Comparison of our proposed method variants and baselines to state of the art on few-shot classification datasets. †: Delta-encoder uses VGG-16 backbone for *mini*ImageNet and CIFAR-FS and ResNet-18 for CUB. Baseline (1), Baseline-KD (2): prototypical classifier at inference, no feature generation. VFH: our vector feature hallucinator; TFH: our tensor feature hallucinator; TFH-ft: our tensor feature hallucinator followed by fine-tuning at inference.

case, the *conditioner* $h : \mathbb{R}^d \to \mathbb{R}^{d'}$ consists of two fully-connected layers with a ReLU activation in-between. The *generator* $g : \mathbb{R}^{k+d'} \to \mathbb{R}^d$ also consists of two fully-connected layers with a ReLU activation in-between and a sigmoid function in the end.

Finally, we experiment with baselines consisting of the embedding network $f_\theta$ (1) or $f_{\theta'}$ (2) at representation learning and a prototypical classifier at inference, without feature hallucination. We refer to them as Baseline (1) and Baseline-KD (2) respectively.

## 3.2 RESULTS

Table 1 compares our method with the state of the art. Most important are comparisons with Chen et al. (2019); Chen et al. (2019); Li et al. (2020); Luo et al. (2021), which use the same backbone, ResNet-18. Our tensor feature hallucinator provides new state of the art performance in all datasets and all settings, outperforming all competing few-shot data augmentation methods. *Fine-tuning* at inference is mostly beneficial, especially at 5-shot tasks. This is expected, as more data means less risk of overfitting. It is clear that the *tensor feature hallucinator* is superior to the vector feature hallucinator, while the latter is still very competitive. *Self-distillation* also provides a significant boost of performance in all experiments.

## 4 CONCLUSION

Our solution is conceptually simple and improves the state of the art of data augmentation methods in the few-shot learning setting. We provided experimental evidence showing that using a simple loss function and exploiting the structural properties of tensors can provide significant improvement in performance. Notably, the importance of using tensor features is evident through comparison with vector features, which are unable to achieve similar performance. Potential future directions include investigating the performance of our method with different backbone architectures and other experimental settings beyond few-shot learning.

## REFERENCES

Mengting Chen, Yuxin Fang, Xinggang Wang, Heng Luo, Yifeng Geng, Xinyu Zhang, Chang Huang, Wenyu Liu, and Bo Wang. Diversity transfer network for few-shot learning. In *AAAI*, 2020.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *ICLR*, 2019.

Z. Chen, Y. Fu, Y. Zhang, Y. Jiang, X. Xue, and L. Sigal. Multi-level semantic feature augmentation for one-shot learning. *IEEE Transactions on Image Processing*, 2019.

Zitian Chen, Yanwei Fu, Yu-Xiong Wang, Lin Ma, Wei Liu, and Martial Hebert. Image deformation meta-networks for one-shot learning. In *CVPR*, 2019.

Nanqing Dong and Eric P Xing. Domain adaption in one-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 573–588. Springer, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

Nathan Hilliard, Lawrence Phillips, Scott Howland, Artëm Yankov, Courtney D Corley, and Nathan O Hodas. Few-shot learning with metric-agnostic conditional embeddings. *arXiv preprint arXiv:1802.04376*, 2018.

Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. Learning to cluster in order to transfer across domains and tasks. *arXiv preprint arXiv:1711.10125*, 2017.

Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *CVPR*, 2019.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML workshop*, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Michalis Lazarou, Yannis Avrithis, and Tania Stathaki. Iterative label cleaning for transductive and semi-supervised few-shot learning. *arXiv preprint arXiv:2012.07962*, 2020.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019.

Kai Li, Yulun Zhang, Kunpeng Li, and Yun Fu. Adversarial feature hallucination networks for few-shot learning. In *CVPR*, 2020.

Xinzhe Li, Qianru Sun, Yaoyao Liu, Qin Zhou, Shibao Zheng, Tat-Seng Chua, and Bernt Schiele. Learning to self-train for semi-supervised few-shot classification. In *NeurIPS*, 2019.

Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.

Qinxuan Luo, Lingfeng Wang, Jingguo Lv, Shiming Xiang, and Chunhong Pan. Few-shot learning via feature hallucination with variational inference. In *WACV*, 2021.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *ICML*, 2017.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *NeurIPS*, 2018.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539*, 2020.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, 2016.

Zhongjie Yu, L. Chen, Zhongwei Cheng, and Jiebo Luo. Transmatch: A transfer-learning scheme for semi-supervised few-shot learning. *CVPR*, 2020.

Hongguang Zhang, Jing Zhang, and Piotr Koniusz. Few-shot learning via saliency-guided hallucination of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2770–2779, 2019.

Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. *NeurIPS*, 2018.

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning.

# A   APPENDIX

## A.1   DATASET DETAILS

***mini*ImageNet**   This is a widely used few-shot image classification dataset Vinyals et al. (2016); Ravi & Larochelle (2016).   It contains 100 randomly sampled classes from ImageNet Krizhevsky et al. (2012). These 100 classes are split into 64 training (base) classes, 16 validation (novel) classes and 20 test (novel) classes. Each class contains 600 examples (images). We follow the commonly used split provided by Ravi & Larochelle (2016).

**CUB**   This is a fine-grained classification dataset consisting of 200 classes, each corresponding to a bird species. We follow the split defined by Chen et al. (2019); Hilliard et al. (2018), with 100 training, 50 validation and 50 test classes.

**CIFAR-FS**   This dataset is derived from CIFAR-100 Krizhevsky et al. (2009), consisting of 100 classes with 600 examples per class. We follow the split provided by Chen et al. (2019), with 64 training, 16 validation and 20 test classes.

All images from all datasets are resized to $224 \times 224$ in a similar way to other data augmentation methods Li et al. (2020); Chen et al. (2019); Chen et al. (2019); Luo et al. (2021)

## A.2   IMPLEMENTATION DETAILS

Our implementation is based on PyTorch Paszke et al. (2017).

**Networks**   In our *tensor feature hallucinator* (TFH), the embedding dimension is $d = 512$ and the resolution $h \times w$ is $7 \times 7$.

The convolutional layers of the *conditioner* use kernels of size $3 \times 3$ and stride 1 and in the input layer we also use padding 1. The channel dimensions are 512 and 256 for the first and second convolutional layers respectively. The dimension of the *class-conditional vector* is set to $d' = 1024$. The tensor dimensions of all conditioner layers are $[512 \times 7 \times 7]$, $[256 \times 5 \times 5]$, $[6400]$ (flattening) and $[1024]$.

All three transpose-convolutional layers of the *generator* use kernels of size $3 \times 3$, stride 1 and 512 channels. The dimension of $z \sim \mathcal{N}(\mathbf{0}, I_k)$ is $k = 1024$. The tensor dimensions of all generator layers are $[2048 \times 1 \times 1]$, $[512 \times 3 \times 3]$, $[512 \times 5 \times 5]$, and $[512 \times 7 \times 7]$.

In our *vector feature hallucinator* (VFH), the dimensions of the *class-conditional vector* as well as the hidden layers of both the *conditioner* and the *generator* are all set to 512.

**Training**   For the *embedding model*, similarly to Tian et al. (2020), we use SGD optimizer with learning rate 0.05, momentum 0.9 and weight decay 0.0005. For data augmentation, as in Lee et al. (2019), we adopt random crop, color jittering, and horizontal flip.

The *tensor feature hallucinator* is trained in a meta-training regime with $N = 5$ classes, $K = 20$ examples per class and generation of $M = 50$ class-conditioned examples in every task. We use Adam optimizer with initial learning rate $10^{-5}$, decaying by half at every 10 epochs. We train for 50 epochs, where each epoch consists of 600 randomly sampled few-shot learning tasks. At test time, we find that generating more class-conditioned examples improves the accuracy, therefore we generate $M = 500$ class-conditioned examples.

Our TFH-ft version uses the novel-class support examples to fine-tune all of its parameters. In the fine-tuning stage, we use exactly the same loss function as in the hallucinator training phase (4) and we fine-tune for 10 steps using Adam optimizer and learning rate of $10^{-5}$.