Tensor feature hallucination for few-shot learning

Michalis Lazarou[1], Tania Stathaki[1] and Yannis Avrithis[2]

[1]Imperial College London, [2]Athena RC

WACV 2022

Virtual, January 2022

Imperial College London

ATHENA Research & Innovation Information Technologies

What is few-shot learning

- Why is it important?

?

Contributions

Previous state of the art

- Meta-learning:
- Transfer learning
- Domain adaptation
- Synthetic data generation

Contributions

- Exploit the spatial properties of tensor features
- Introduce a novel loss function that is simpler than the other state of the art
- Provide state of the art performance in three benchmark datasets

Contributions

Previous state of the art

- Meta-learning:
- Transfer learning
- Domain adaptation
- Synthetic data generation

Contributions

- Exploit the spatial properties of tensor features
- Introduce a novel loss function that is simpler than the other state of the art
- Provide state of the art performance in three benchmark datasets

Problem formulation and definitions

Representation learning

- Base class dataset: $D_{\text{base}} := \{(x_i, y_i)\}_{i=1}^{I}$ where $y_i \in C_{\text{base}}$
- Train embedding network $f_\theta : \mathcal{X} \to \mathbb{R}^{d \times h \times w}$ on D_{base}
- $\bar{f}_\theta : \mathcal{X} \to \mathbb{R}^d$ denotes f_θ followed by global average pooling (GAP)

Inference stage

- Novel class dataset D_{novel} with C_{novel} disjoint from C_{base}
- Sample a support set S and a query set Q from D_{novel}
- S consists of N classes with K labeled examples per class
- Given S and f_θ, classify examples from Q

Problem formulation and definitions

Representation learning

- Base class dataset: $D_{\text{base}} := \{(x_i, y_i)\}_{i=1}^{I}$ where $y_i \in C_{\text{base}}$
- Train embedding network $f_\theta : \mathcal{X} \to \mathbb{R}^{d \times h \times w}$ on D_{base}
- $\bar{f}_\theta : \mathcal{X} \to \mathbb{R}^d$ denotes f_θ followed by global average pooling (GAP)

Inference stage

- Novel class dataset D_{novel} with C_{novel} disjoint from C_{base}
- Sample a support set S and a query set Q from D_{novel}
- S consists of N classes with K labeled examples per class
- Given S and f_θ, classify examples from Q

Representation learning

- 1st stage: train the backbone network using standard cross-entropy

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

  2nd stage: adopt a self-distillation process

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

Meta-training tensor hallucinator

- Meta-training stage: sample episodes of S from D_{base}
- Pre-trained network $f_\theta : \mathcal{X} \to \mathbb{R}^{d \times h \times w}$ maps images to tensors
- Class prototype tensors:

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

- Conditioner h maps prototypes to class-conditional vectors $s_j := h(p_j) \in \mathbb{R}^{d'}$
- Generator maps to $g(z; s_j) \in \mathbb{R}^{d \times h \times w}$ where $z \sim \mathcal{N}(\mathbf{0}, I_k)$
- Jointly trained by:

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

Meta-training tensor hallucinator

- Meta-training stage: sample episodes of S from D_{base}
- Pre-trained network $f_\theta : \mathcal{X} \to \mathbb{R}^{d \times h \times w}$ maps images to tensors
- Class prototype tensors:

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

- Conditioner h maps prototypes to class-conditional vectors $s_j := h(p_j) \in \mathbb{R}^{d'}$
- Generator maps to $g(z; s_j) \in \mathbb{R}^{d \times h \times w}$ where $z \sim \mathcal{N}(\mathbf{0}, I_k)$
- Jointly trained by:

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

Meta-training tensor hallucinator

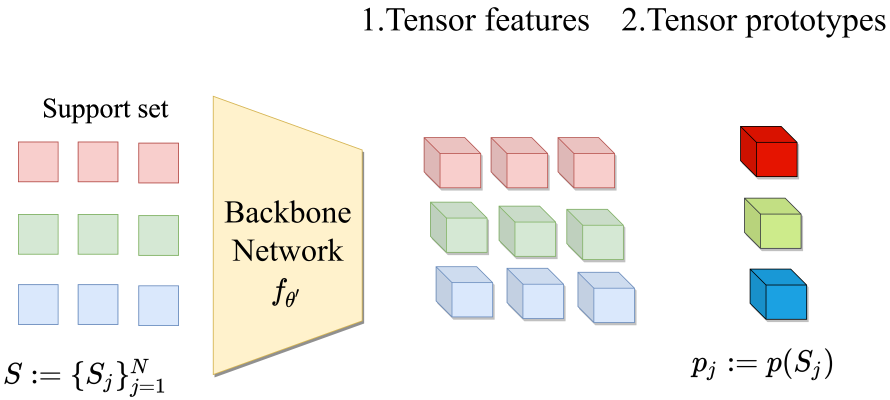
- Meta-training stage: sample episodes of S from D_{base}
- Pre-trained network $f_\theta : \mathcal{X} \to \mathbb{R}^{d \times h \times w}$ maps images to tensors
- Class prototype tensors:

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

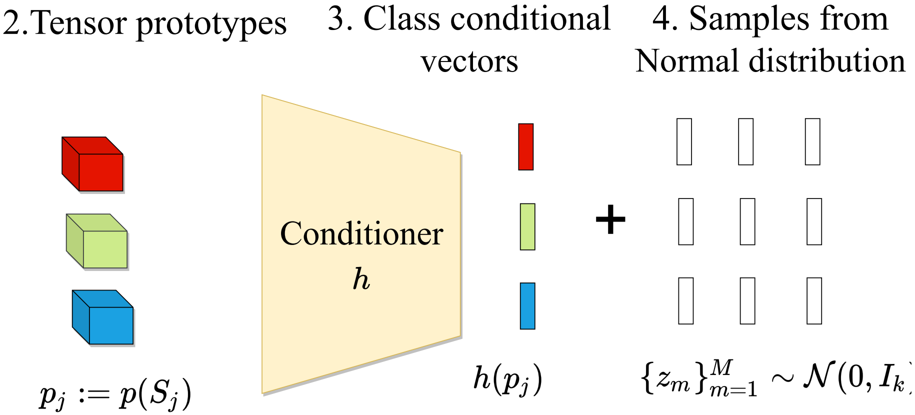
- Conditioner h maps prototypes to class-conditional vectors $s_j := h(p_j) \in \mathbb{R}^{d'}$
- Generator maps to $g(z; s_j) \in \mathbb{R}^{d \times h \times w}$ where $z \sim \mathcal{N}(\mathbf{0}, I_k)$
- Jointly trained by:

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

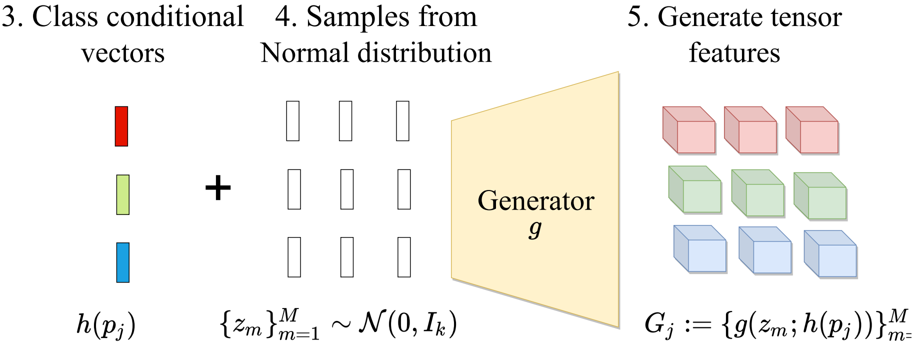
Tensor hallucinator: inference



Tensor hallucinator: inference



Tensor hallucinator: inference

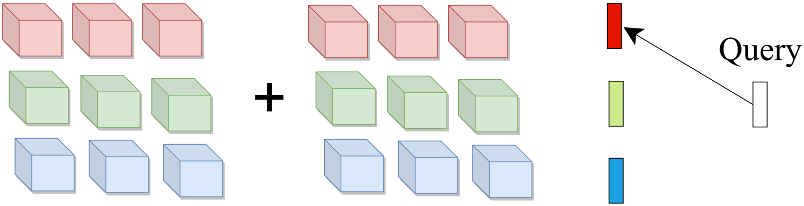


Tensor hallucinator: inference

5. Generate tensor features

6. Augment Support tensors

7. GAP and prototype classification



$$G_j := \{g(z_m; h(p_j))\}_{m=1}^{M}$$

Experimental results

Method	*mini*ImageNet		CUB	
	1-shot	5-shot	1-shot	5-shot
Dual TriNet [Chen *et al.* 2019]	58.80±1.37	76.71±0.69	69.61	84.10
IDeMe-Net [Chen *et al.* 2019]	59.14±0.86	74.63±0.74	–	–
AFHN [Li *et al.* 2020]	62.38±0.72	78.16±0.56	70.53±1.01	83.95±0.63
VI-Net [Luo *et al.* 2021]	61.05	78.60	74.76	86.84
Baseline (1)	56.81±0.81	78.31±0.59	67.14±0.89	86.22±0.50
Baseline-KD (2)	59.62±0.85	79.31±0.62	70.85±0.90	87.64±0.48
VFH (ours)	61.88±0.85	79.63±0.61	75.44±0.85	87.82±0.47
TFH (ours)	64.49±0.84	79.94±0.60	75.66±0.85	88.39±0.49
TFH-ft (ours)	**65.07**±0.82	**80.81**±0.61	**75.76**±0.83	**88.60**±0.47

Experimental results

METHOD	*mini*IMAGENET		CUB	
	1-shot	5-shot	1-shot	5-shot
Dual TriNet [Chen *et al.* 2019]	58.80±1.37	76.71±0.69	69.61	84.10
IDeMe-Net [Chen *et al.* 2019]	59.14±0.86	74.63±0.74	–	–
AFHN [Li *et al.* 2020]	62.38±0.72	78.16±0.56	70.53±1.01	83.95±0.63
VI-Net [Luo *et al.* 2021]	61.05	78.60	74.76	86.84
Baseline (1)	56.81±0.81	78.31±0.59	67.14±0.89	86.22±0.50
Baseline-KD (2)	59.62±0.85	79.31±0.62	70.85±0.90	87.64±0.48
VFH (ours)	61.88±0.85	79.63±0.61	75.44±0.85	87.82±0.47
TFH (ours)	64.49±0.84	79.94±0.60	75.66±0.85	88.39±0.49
TFH-ft (ours)	**65.07**±0.82	**80.81**±0.61	**75.76**±0.83	**88.60**±0.47

Thank you!

https://github.com/MichalisLazarou
https://avrithis.net
http://www.commsp.ee.ic.ac.uk/~tania/