Localized Latent Updates for Fine-Tuning Vision-Language Models

Moritz Ibing Isaak Lim Leif Kobbelt
Visual Computing Institute, RWTH Aachen University

Abstract

Although massive pre-trained vision-language models like CLIP show impressive generalization capabilities for many tasks, still it often remains necessary to fine-tune them for improved performance on specific datasets. When doing so, it is desirable that updating the model is fast and that the model does not lose its capabilities on data outside of the dataset, as is often the case with classical fine-tuning approaches. In this work we suggest a lightweight adapter that only updates the models predictions close to seen datapoints. We demonstrate the effectiveness and speed of this relatively simple approach in the context of few-shot learning, where our results both on classes seen and unseen during training are comparable with or improve on the state of the art.

1. Introduction

Much of the success of deep learning when it comes to vision tasks, such as classification, object detection or segmentation, is due to ever bigger models trained on increasingly large quantity of data.

A popular approach to make use of immense sources of uncurated data in the form of images with textual descriptions are vision-language models [20, 27]. Here both image and description are individually mapped into a joint embedding space. This embedding is optimized, so that matching pairs are close, and the distance between all other pairs large. A model trained in this fashion can be used for zero-shot classification, as the language model can deal with every conceivable class by embedding a textual description (*e.g.* "a picture of [CLASS]").

Even though these models can be applied to all kinds of classification tasks, their performance sometimes is suboptimal. This might be the case if used on a dataset with specific characteristics that differ from the original training set, *e.g.* if the task is to recognize the action performed in an image, even though during training the model only saw generic objects. In these cases a common technique is to fine-tune the pre-trained model for the task at hand. However, updating the complete model is quite expensive (as the employed models are large). There are two solutions proposed in the literature to tackle this problem. One is prompt-learning [38–40] where the context around the class ("a picture of" in the last example) is optimized for a specific dataset instead of hand crafted. The other option is to use adapters [12, 37], light-weight models (usually small MLPs) that modify the embedding produced by either the visual or language model (or both), thus updating the predictions without the need to update the original networks parameters.

Both these approaches however still have the problem that even though the performance is improved for the specific domain and task the fine-tuning was done for, this comes at the cost of a decrease in performance on other tasks/domains compared to the original model [13].

The goal of this work is to reap the benefits of fine-tuning on a specific task, without losing the generalization ability of the original model. Work in this direction has already been done in the form of CoCoOp [38], where prompt-learning is employed, but the context is not fine-tuned on a specific dataset, but instead a suitable context is predicted from the image to be classified. Another approach using prompt-learning is ProGrad [40] where the context update is restricted in order not to lose information from the pre-training stage. Although both methods decrease the performance loss in the zero-shot setting, they still do not reach the abilities of the original model.

In contrast we choose a simple method based on adapters. The idea is to only update the embedding where we actually have training data and leave it unchanged everywhere else, thus retaining the original predictions of the model, where we cannot improve on them. Furthermore, even where we have data we want to change the embedding as little as possible, to allow sensible interpolation between fine-tuned and original embedding. This approach is extremely lightweight, as we only need to tune a small amount of parameters, and back-propagating through the original model is not necessary. Still we show an improvement in performance compared to the previous state of the art.

2. Related Work

Zero-Shot Learning Zero-shot learning describes a setting, where the set of classes at training and during testing are disjoint or at least not identical [4], thus the relation between classes and images belonging to that class needs to be learned indirectly. There are numerous works of research in this area, so we instead refer to [33, 35] for an overview. A common approach to tackle this task is to relate pre-trained image and class embeddings [1, 2, 11]. This is conceptually very close to vision-language models, another popular framework that can be applied for zero-shot learning.

Vision-Language Models The term vision-language model describes networks that learn an alignment between images and text in a joint embedding space. A large amount of work has been done in this area [9, 11, 22, 29], usually based on contrastive learning, which has been popularized for pre-training of image models [5,14,17] and aims to maximize the distance between similar and dissimilar instances in the embedding space. Currently one of the most popular vision-language models is CLIP [27], which we use in all our experiments. It uses a Transformer [31] as text encoder and a ResNet [15] or ViT [8] as image encoder. ALIGN [20] is a similar approach, whereas DeCLIP [23] tries to improve the training procedure in order reach the same performance with less data.

Fine-Tuning When it comes to fine-tuning a pre-trained vision-language model, there are broadly three types of approaches in the literature. It is possible to fine-tune the entire model, but afterwards interpolate between the original and updated weights, to counteract overfitting (WiSE-FT [34]). Alternatively, not the model itself is trained, but only an adapter that is applied onto the embedding space (CLIP-Adapter [12]). This is the approach we choose as well. Instead of learning this adapter, it can be extracted from the fine-tuning dataset (TIP-Adapter [37]). As this requires data for every class it is evaluated on, it is however not suitable for zero-shot learning. The last approach is called prompt engineering. Here the context of the text embedding is optimized for performance on the training set (CoOp [39]). This learning can be restricted in order not to decrease the loss of information from the pre-training stage (ProGrad [40]). Another option is to predict the (textual) context for each image (CoCoOp [38]). This mitigates overfitting on the train set and thus is better at retaining zero shot ability on unseen classes, but comes at the cost of an increased training time.

3. Method

Before introducing our approach in more detail, we will give a short overview on how vision-language models work in general on the example of CLIP, which is used in all our experiments.

3.1. Vision-Language Models

Vision-Language models consist of two networks: an image encoder f_I and a text encoder f_T. Their exact implementation is of no interest to us in this context. All we need to know, is that these models embed an image or a text respectively to a (normalized) feature vector of the same dimension. The cosine distance between an embedded image and text should then correspond to their similarity *i.e.* how well the text describes the image.

During training, we are given a batch of n images X and their textual descriptions Y. We make the simplifying assumption that each text is a perfect description of the corresponding image and all other texts are completely unrelated. Thus, we want to minimize the cosine distance between embeddings of matching image/text pairs $f_I(x_i)$,$f_T(y_i)$ and maximize the distance between all other pairs within the batch $f_I(x_i)$,$f_T(y_j)$ with $i \neq j$.

Another view would be to regard the cosine distance as the likelihood that a given text y describes the corresponding image x, or vice versa. We can compute the normalized probabilities as:

$$p(y|x) = \frac{\exp(f_I(x)^T f_T(y)/\tau)}{\sum_{i=1}^{n} \exp(f_I(x)^T f_T(y_i)/\tau)} \tag{1}$$

where τ is a learned temperature parameter. As we assume the embeddings to be normalized, the dot product is equivalent to the cosine similarity. The probability $p(x|y)$ only differs in the normalization.

In this view it now makes sense to maximize the probability for the correct pairs, for which we can use the Cross Entropy loss. As we want to maximize the probabilities in both directions the loss is given as:

$$L = -\frac{1}{N} \sum_{i=1}^{N} (\log(p(x_i|y_i)) + \log(p(y_i|x_i)))/2 \tag{2}$$

Zero-Shot Classification This approach leads to a semantically meaningful embedding of both images and text that can be used for downstream tasks. Alternatively, we can use it directly for zero-shot classification. For this we embed a text for each class (*e.g.* a picture of [CLASS]) to a feature vector $f_T(y_k)$. Then, to classify a given image, we embed it and compute the distance between its feature vector $f_I(x)$ and all class embeddings. The class probability for a class k is then given similarly as before as $p(y_k|x)$.

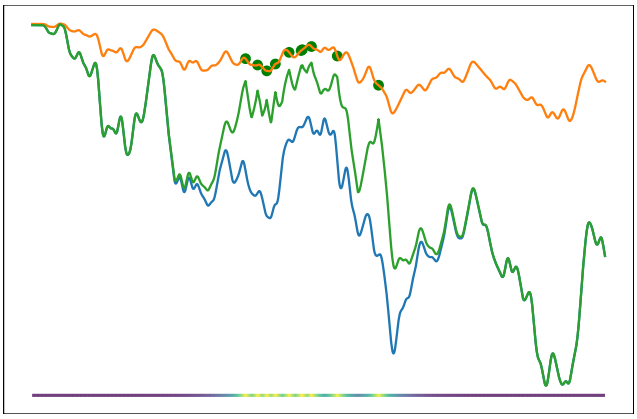

Figure 1. The basic idea of our approach is to interpolate between the original networks output (blue) and the newly trained adapter (orange) to obtain the final function (green). The interpolation weight (bottom line) is based on the distance to the samples we fine-tuned on (green points).

3.2. Local Linear Updates

A big benefit of vision-language models is their generalization capability. As they are usually trained on immense datasets, they tend to show great zero-shot capabilities even on unseen domains. However, they are not necessarily well tuned for small specific datasets that were not well represented in the original training data.

Although we could fine-tune the networks on this specific set, this would come at the cost of the models ability to generalize and be quite expensive. Instead we add additional functions on the output of the model $g_I \circ f_I$ and $g_T \circ f_T$ respectively and optimize only those. This is much cheaper, as g has much less parameters (we will omit the subscript for both f and g if both the image and textual models are meant). Furthermore, as g is applied onto the output of f we do not even have to back-propagate through the big models. We could even precompute f on the given dataset to save further computation cost. In our case we use distinct networks for g_I and g_T but is possible to use the same function as well ($g_I = g_T$) as they are applied on the same domain.

The training works slightly differently than before. As we now do not have unique image/text pairs, but instead images with their classes, we leave out one half of the loss, leading to the classical Cross Entropy loss:

$$L = -\frac{1}{N}\sum_{i=1}^{N} \log(p(y_i|x_i)) \tag{3}$$

A similar idea was presented in CLIP Adapter [12] (Here only g_I is used). This approach significantly reduces the cost of fine-tuning, but does not solve the problem of loosing capabilities on the previous training dataset while overfitting on the new one.

Local Interpolation To lessen the overfitting WISE-FT [34] interpolates the weights between the original model and a fine-tuned version. With a similar aim in CLIP Adapter [12] the results of f and g are interpolated:

$$\alpha(g \circ f) + (1-\alpha)f \tag{4}$$

Here α is a global parameter. It would be more sensible to localize this interpolation to the area of the feature space, where we obtained new data. Only there do we actually have information onto how to sensibly update the embeddings. As g is a global function but we only supervise it at our training samples, it is unlikely that it represents a sensible modification away from these samples. Thus, in our case α is not a global parameter, but a function,

$$\alpha(x, D) = \beta \cdot \max_{d \in D}(\exp(-\gamma(1 - x^T d))) \tag{5}$$

where D is the set of datapoints we fine-tuned on and β is a global parameter similar to how α was defined previously. γ concentrates the focus of the interpolation mask and thus influences in what range our updated embedding should be applied. If we would let it go towards zero, we would have a global α parameter, similarly to CLIP Adapter. On the other hand, if we let it go towards infinity, we would only update exactly the images and classes on which we fine-tuned.

Whenever an image is close to one already seen during training, we thus use our updated features, otherwise we utilize the general knowledge of the pre-trained model. Note that we do this separately for the text and image encoder, so there are separate sets D_{text} and D_{image}.

Clustering This approach requires us to save the feature vectors of all datapoints seen during fine-tuning. This is not a problem, as long as the dataset used is indeed very small. If this however is not the case, we cluster the feature vectors to find sensible representatives. For this we use agglomerative clustering, where we start by regarding each datapoint as an individual cluster and then iteratively merge pairs based on the maximum distance between their members until we have reached the desired number of clusters. Each cluster needs a position, which is computed as the (normalized) mean of all its members.

We chose this approach, as it does not make any assumptions about cluster shapes nor does it require a sensible initialization. Furthermore, we expect the number of clusters to be in a similar order of magnitude as the number of datapoints, so we do not need many merge operations. However, as we show in the ablation (Sec.4.4), clustering only has a small effect on the performance and is mainly used to bound the memory consumption. Thus, the exact clustering algorithm is not likely to make much of a difference either.

Identity Regularization So far, we have restricted the region, where we change the feature space, but not the magnitude of the update, which can become arbitrarily large. It is however desirable that the update is as small as possible, while minimizing the training loss. As we assume the original pre-trained features to already be useful, we want to stay as close to them as possible in order to retain generality. Furthermore the interpolation between f and $g \circ f$ should result in sensible embeddings, which is more likely the case, if they are close to each other. In other words, g should stay as close to the identity as possible.

This is easy to enforce, if we simply choose g as an affine function $g = Wx + b$. In this case our regularization takes the form:

$$\lambda(||W - I||_2 + ||b||_2) \tag{6}$$

where I is the identity matrix and λ a weighting parameter.

Of course we could choose a more complex function and regularize g to stay close to the identity at a set of sample points. However, a dense sampling of the embedding space is infeasible, and we are interested in retaining this property wherever the interpolation weight α is non-zero.

Furthermore, we initialize g as the identitiy function, which is trivial for affine functions, but not for non-linear MLPs.

4. Evaluation

For our evaluation we follow CoCoOp [38], where three problem settings are investigated:

1. Generalization to new classes *within* a given dataset.
2. Generalization to new datasets after fine-tuning.
3. Generalization to domain-shift.

Before presenting the conclusions, we will introduce the used datasets, and explain the training procedure.

Datasets Similar to CoCoOp, we follow CoOp [39] in the choice of datasets used in evaluation. To be precise we use 11 datasets that cover a wide range of tasks: ImageNet [7] and Caltech101 [10] for generic object classification, OxfordPets [26], StanfordCars [21], Flowers102 [25], Food101 [3] and FGVCAircraft for more specific object classification, [24], SUN397 [36], DTD [6], EuroSAT [16] and UCF101 [30] for a diverse set of tasks. Furthermore, to evaluate domain generalization we regard ImageNet as source and four different versions under different types of domain shift as target. The four datasets are: ImageNetV2 [28], ImageNet-Sketch [32], ImageNet-A [19] and ImageNet-R [18].

The set of images for few-shot training are randomly sampled for each dataset, while using the original test set for testing. For approaches that need training we average the results over three runs.

Training Our implementation is based on the published code of CoOp. We use the same learning rate and number of epochs as they do. Following CoCoOp we use ViT-B/16 as the vision backbone of CLIP. As ProGrad has been evaluated on a different backbone, we retrained it for a fair comparison. Note that both CoOp and CoCoOp have a context length of 4 initialized as the prompt: "a photo of a", whereas for CLIP Adapter and us the context is class dependent and ProGrad has a context length of 16 with a class-dependent initialization. If not stated otherwise we choose the parameters of our approach as $\beta = 0.5$, $\gamma = 20$, $\lambda = 1e3$ and the number of clusters as 512.

4.1. Base to New Generalization

On each dataset, the classes are split equally into a set of "base" classes on which the adapter is trained and unseen "new" classes, where we only evaluate. Thus, no matter how many shots are given for the training, on the new classes we will always do zero shot inference. We show results for different numbers of shots in Figure 2 and report exact values in Table 1. Here we also give the harmonic mean between the evaluation on base and new classes for an easier comparison of the approaches regarding their respective trade-offs.

As can be seen, on the 16 shot evaluation our approach outperforms all other methods on 8 out of 11 datasets, when regarding the harmonic mean. Here we have on average an improvement of almost 3 percentage points to CoCoOp (the next best method). Furthermore, (on average) our localized adapter beats all other methods regarding new classes independent of the number of shots and is first or second on the base classes.

Our method is the only one that reaches the performance of CLIP when it comes to zero-shot performance on unseen classes, whereas all other methods show a drop in performance here that usually increases with the number of shots, hinting at overfitting.

4.2. Cross-Dataset Generalization

In this experiment the models are fine-tuned on ImageNet and then evaluated on the other datasets, thus an improvement on ImageNet (compared to CLIP) is expected. Interestingly both CoCoOp and our approach show an improvement (on average) on the other datasets as well. Apparently the training samples of ImageNet are numerous and diverse enough to avoid overfitting and the data distribution of ImageNet is closer to the other regarded datasets than the original training set of CLIP. Although our method does not reach the results of CoCoOp on this evaluation we come very close.

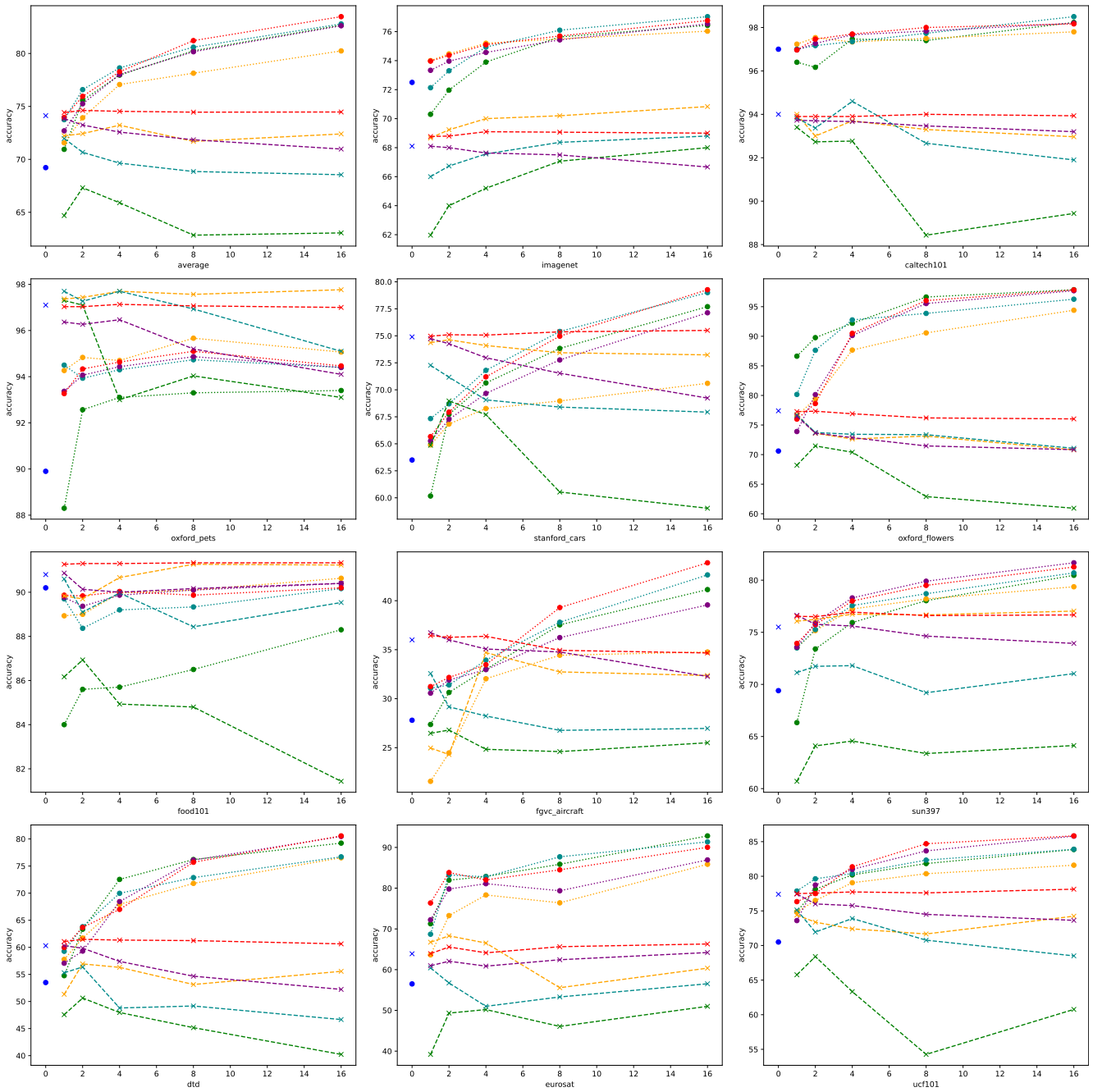

Figure 2. Comparison in the intra-class generalization setting. We compare our approach (red) vs. CoOp (green), CoCoOp (yellow), ProGrad (cyan) and CLIP Adapter (purple). Zero-shot CLIP is shown as a baseline in blue. circles mark the base classes and x the unseen new classes.

4.3. Domain Generalization

In this last comparison the models are again fine-tuned on ImageNet and then evaluated on different versions with a clear domain shift. Here we can see a slight drop of performance between our method and prompt based approaches. This might be due to the fact that prompt-based approaches only fine-tune the input of the text encoder. As the class names and thus the text encodings are not affected by domain shift, their performance generalizes better. On the other hand we directly update the text and image embedding (and CLIP Adapter only updates the image embedding), which might be problematic as here changes caused

	Base	New	H
CLIP	69.34	74.22	71.70
CoOp	82.69	63.22	71.66
CoCoOp	80.47	71.69	75.83
ProGrad	82.79	68.55	74.46
CLIP Ada.	82.62	70.97	76.02
LLU	**83.48**	**74.47**	**78.46**

(a) Average over 11 datasets

	Base	New	H
CLIP	72.43	68.14	70.22
CoOp	76.47	67.88	71.92
CoCoOp	75.98	**70.43**	**73.10**
ProGrad	**77.03**	68.8	72.68
CLIP Ada.	76.53	66.67	71.26
LLU	76.77	69.00	72.68

(b) ImageNet

	Base	New	H
CLIP	96.84	**94.00**	95.40
CoOp	98.00	89.81	93.73
CoCoOp	97.96	93.81	95.84
ProGrad	**98.50**	91.90	95.09
CLIP Ada.	98.20	93.20	95.63
LLU	98.17	93.93	**96.00**

(c) Caltech101

	Base	New	H
CLIP	91.17	97.26	94.12
CoOp	93.67	95.29	94.47
CoCoOp	**95.20**	**97.69**	**96.43**
ProGrad	94.40	95.10	94.75
CLIP Ada.	94.40	94.10	94.25
LLU	94.47	97.00	95.72

(d) OxfordPets

	Base	New	H
CLIP	63.37	74.89	68.65
CoOp	78.12	60.40	68.13
CoCoOp	70.49	73.59	72.01
ProGrad	79.00	67.93	73.05
CLIP Ada.	77.13	69.23	72.97
LLU	**79.27**	**75.50**	**77.34**

(e) StanfordCars

	Base	New	H
CLIP	72.08	**77.80**	74.83
CoOp	97.60	59.67	74.06
CoCoOp	94.87	71.75	81.71
ProGrad	96.27	71.07	81.77
CLIP Ada.	97.70	70.83	82.13
LLU	**97.83**	76.03	**85.57**

(f) Flowers102

	Base	New	H
CLIP	90.10	91.22	90.66
CoOp	88.33	82.26	85.19
CoCoOp	**90.70**	91.29	**90.99**
ProGrad	90.17	89.53	89.85
CLIP Ada.	90.40	90.40	90.40
LLU	90.20	**91.33**	90.76

(g) Food101

	Base	New	H
CLIP	27.19	**36.29**	31.09
CoOp	40.44	22.30	28.75
CoCoOp	33.41	23.71	27.74
ProGrad	42.63	26.97	33.04
CLIP Ada.	39.57	32.27	35.55
LLU	**43.87**	34.67	38.72

(h) FGVCAircraft

	Base	New	H
CLIP	69.36	75.35	72.23
CoOp	80.60	65.89	72.51
CoCoOp	79.74	**76.86**	78.27
ProGrad	80.70	71.03	75.56
CLIP Ada.	**81.67**	73.93	77.61
LLU	81.27	76.67	**78.90**

(i) SUN397

	Base	New	H
CLIP	53.24	59.90	56.37
CoOp	79.44	41.18	54.24
CoCoOp	77.01	56.00	64.85
ProGrad	76.70	46.67	58.03
CLIP Ada.	80.47	52.23	63.35
LLU	**80.56**	**60.63**	**69.19**

(j) DTD

	Base	New	H
CLIP	56.48	64.05	60.03
CoOp	**92.19**	54.74	68.69
CoCoOp	87.49	60.04	71.21
ProGrad	91.37	56.53	69.85
CLIP Ada.	86.93	64.20	73.86
LLU	90.33	**66.30**	**76.37**

(k) EuroSAT

	Base	New	H
CLIP	70.53	77.50	73.85
CoOp	84.69	56.05	67.46
CoCoOp	82.33	73.45	77.64
ProGrad	83.90	68.50	75.42
CLIP Ada.	85.80	73.63	79.25
LLU	**85.83**	**78.13**	**81.80**

(l) UCF101

Table 1. Comparison in the intra-class generalization setting. All methods except for CLIP (CoOp, CoCoOp, ProGrad, CLIP Adapter and our method LLU (Localized Linear Updates)) are trained on the base classes with 16 shots. H denotes the harmonic mean.

by the domain-shift have a more direct effect.

4.4. Further Analysis

Ablation Here we discuss the effect different design choices in our approach have on the result. For this we regard the average performance achieved on the Base to New training setup, when using 16 shots (Table 4). As already mentioned, it barely makes any difference, whether we use clustering or not ("no cluster"), thus it is a sensible choice to limit the memory requirements. Using the global dampening parameter β does lead to an improvement, although it is rather small ("no damp").

Not restricting the interpolation to the training samples ("no mask") leaves the results on the base classes unchanged but leads to overfitting and thus a reduced performance on unseen classes. On the other hand focusing the interpolation exclusively on the training samples ("Dirac mask" equivalent to a γ parameter of infinity) leads to the same performance as CLIP for new classes, but of course this way our method cannot improve on unseen classes either, as seen in Tables 2 and 3. Note that for numerical reasons, we did not actually implement a γ parameter of infinity, but clamped α to zero or one, depending on some small distance threshold.

Interestingly leaving out the identity regularization ("no reg") barely has any effect on the results, whereas an initial-

	Source	Target										
	ImageNet	Caltech101	OxfordPets	StanfordCars	Flowers102	Food101	FGCVAircraft	SUN397	DTD	EuroSAT	UCF101	Average
CLIP	66.7	92.8	89.2	65.3	68.2	85.8	**24.9**	62.6	44.5	47.7	66.7	64.77
CoOp	71.51	93.70	89.14	64.51	68.71	85.30	18.47	64.15	41.92	46.39	66.55	63.88
CoCoOp	71.02	**94.43**	**90.14**	65.32	**71.88**	86.06	22.94	**67.36**	**45.73**	45.37	**68.21**	**65.74**
ProGrad	72.00	92.67	89.73	64.00	68.37	85.27	20.30	64.60	43.07	44.53	65.20	63.78
CLIP Adapter	71.77	92.17	86.47	60.50	67.63	82.53	22.90	62.77	42.23	47.67	63.37	62.82
LLU	**72.13**	92.00	89.10	**65.37**	71.23	**86.10**	24.87	64.93	44.63	**47.77**	67.10	65.31

Table 2. Comparison for cross dataset generalization capability. All approaches are trained on ImageNet (16 shots) and then evaluated on all 11 datasets.

	Source	Target			
	ImageNet	ImageNetV2	ImageNet-Sketch	ImageNet-A	ImageNet-R
CLIP	66.73	60.83	46.15	47.77	73.96
CoOp	71.51	64.20	47.99	49.71	75.21
CoCoOp	71.02	64.07	**48.75**	**50.63**	**76.18**
ProGrad	72.00	**64.70**	48.37	49.73	75.57
CLIP Adapter	71.77	63.97	46.27	47.80	72.10
LLU	**72.13**	64.53	47.17	48.87	74.30

Table 3. Comparison for domain generalization capability. All approaches are trained on the standard version of ImageNet (16 shots) and then evaluated on 4 different types of domain shift.

ization to identity seems to be more important ("rand. init"). We assume that the training does not include enough update steps for effects to be seen. To substantiate this claim we report the actual distance to identity in Table 4 (for relevant experiments). Here we can see that the distance between our adapter and the identity function does correlate with performance. Lastly we can see that only using a single Linear Layer as an adapter without any of our additional improvements leads to significantly worse results, especially on the unseen classes, thus each of our improvements makes only a small difference individually, but together they significantly increase performance.

Training speed A comparison of the training speed between our approach and prompt based methods depends on both the batch size and the number of classes.

As we can precompute the class embeddings, the training time of our method is almost independent of their number. Prompt based approaches instead need to compute the class embeddings in every iteration. On the other hand the number of class embeddings is independent of the batch size, whereas our adapter needs to be applied to every training sample. In Figure 3 we show the timing for a single forward and backward pass depending on the batch size. As can be seen our method and CLIP Adapter are consistently the fastest and the difference in their timings is negligible (it is barely possible to differentiate their lines). Gener-

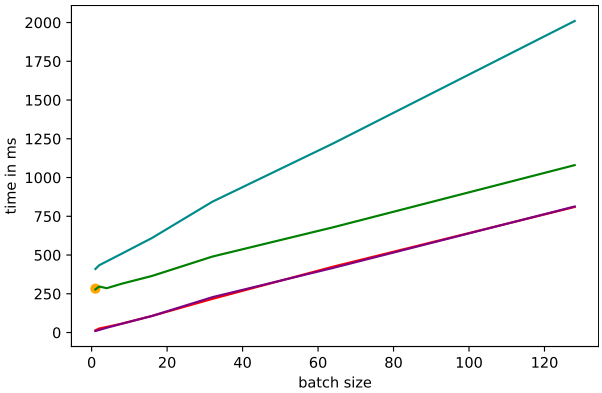

Figure 3. Comparison of timings. Our approach (red), CoOp (green), ProGrad (cyan) and CLIP Adapter (purple). CoCoOp is marked as an orange dot, as a batch size bigger than one does not fit into memory

	CLIP	Default	no cluster	no damp.	no mask	Dirac mask	no reg	rand. init	Linear
Base	69.34	83.48	83.35	83.58	83.41	83.28	83.32	82.91	81.02
New	74.22	74.47	74.40	73.98	72.53	74.21	74.42	72.67	32.60
Mean	71.70	78.46	78.36	78.16	77.27	78.22	78.33	76.94	45.79
Regularization	-	1.95e-5	-	-	-	-	5e-4	3.6e-3	6.5e-3

Table 4. Ablation of different design decisions in our network. The details of the different experiments are explained in subsection 4.4

ally the overhead of the computations due to the adapter barely matter, as can be discerned from the similar slope of CoOp and the adapter based methods. The number of classes influences the distance between these parallel lines, which signifies the overhead due to the computation of their embedding.

For CoCoOp we only have a single data point, as batches with more than one sample do not fit into memory. Thus, although for a batch size of one the training speed is similar to CoOp, in practice CoCoOp is much slower, as we cannot increase the batch size. ProGrad is consistently slower than other methods due to additional computations needed for gradient decomposition.

5. Conclusion

As the requirements in size, data and compute for state of the art AI models increases, it becomes more and more important to be able use available pre-trained networks for complex downstream tasks. In order to do this we need to be able to fine-tune these models in an efficient manner, preferably without loosing the generalization capability that makes them so useful in the first place.

We have introduced an extremely simple approach for this task, introducing small linear updates to the embedding space, localized to the datapoints, where we fine-tune. Our model is fast to train and needs a minimal amount of extra parameters, but still reaches state of the art results both on fine-tuned and unseen classes.

In this work we always trained our adapter for optimal performance on a single dataset. A possible future research direction would be to generalize our approach to multiple distinct fine-tuning datasets. It would be possible to use dataset-dependent adapters and interpolation weights, but some further work would be needed to make this scalable.

Acknowledgements This work was funded by the German Research Foundation within the Gottfried Wilhelm Leibniz programme, as well as through the project "Training the Archive" in cooperation with the Ludwig Forum Aachen and the HMKV Hartware MedienKunstVerein, Dortmund. "Training the Archive" is funded by the Digital Culture Programme of the Kulturstiftung des Bundes (German Federal Cultural Foundation). Funded by the Beauftragte der Bundesregierung für Kultur und Medien (Federal Government Commissioner for Culture and the Media). We would like to thank Dominik Bönisch (heading the "Training the Archive" project) for helpful discussions.

References

[1] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *IEEE TPAMI*, 38(7):1425–1438, 2015. 2

[2] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2927–2936, 2015. 2

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pages 446–461. Springer, 2014. 4

[4] Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *European conference on computer vision*, pages 52–68. Springer, 2016. 2

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2

[6] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014. 4

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 4

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 2

[9] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. In *BMVC*, 2018. 2

[10] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In

2004 conference on computer vision and pattern recognition workshop, pages 178–178. IEEE, 2004. 4

[11] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. *Advances in neural information processing systems*, 26, 2013. 2

[12] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *arXiv preprint arXiv:2110.04544*, 2021. 1, 2, 3

[13] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013. 1

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[16] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019. 4

[17] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International conference on machine learning*, pages 4182–4192. PMLR, 2020. 2

[18] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021. 4

[19] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021. 4

[20] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021. 1, 2

[21] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 4

[22] Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, et al. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *Proceedings of the IEEE international conference on computer vision*, pages 4247–4255, 2015. 2

[23] Yangguang Li, Feng Liang, Lichen Zhao, Yufeng Cui, Wanli Ouyang, Jing Shao, Fengwei Yu, and Junjie Yan. Supervision exists everywhere: A data efficient contrastive language-image pre-training paradigm. In *International Conference on Learning Representations*, 2022. 2

[24] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 4

[25] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. 4

[26] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012. 4

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 1, 2

[28] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019. 4

[29] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. *Advances in neural information processing systems*, 26, 2013. 2

[30] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 4

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[32] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. *Advances in Neural Information Processing Systems*, 32, 2019. 4

[33] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37, 2019. 2

[34] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971, 2022. 2, 3

[35] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4582–4591, 2017. 2

[36] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer so-*

ciety conference on computer vision and pattern recognition, pages 3485–3492. IEEE, 2010. 4

[37] Renrui Zhang, Rongyao Fang, Peng Gao, Wei Zhang, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. *arXiv preprint arXiv:2111.03930*, 2021. 1, 2

[38] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16816–16825, 2022. 1, 2, 4

[39] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, pages 1–12, 2022. 1, 2, 4

[40] Beier Zhu, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang. Prompt-aligned gradient for prompt tuning. *arXiv preprint arXiv:2205.14865*, 2022. 1, 2