# Patching open-vocabulary models by interpolating weights

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

While open-vocabulary models like CLIP achieve high accuracy across many image classification tasks, there are still settings where their zero-shot performance is far from optimal. We study *model patching*, where the goal is to improve accuracy on specific tasks (i.e., patching tasks) without degrading accuracy on tasks where performance is already adequate (i.e., supported tasks). Given a task to be patched, we first fine-tune without introducing new parameters, then interpolate the fine-tuned model weights with the model weights before fine-tuning. We explore model patching on nine tasks where zero-shot CLIP performs poorly, observing that patching increases accuracy by 15 to 60 percentage points while preserving accuracy on ImageNet within one percentage point of the zero-shot model. Additionally, we find that patching is more effective for larger models, demonstrate that a single model can be patched on multiple tasks, and identify cases of broad transfer, where patching one one task can increase accuracy on other tasks even when the classes are not shared. Finally, we investigate applications beyond standard benchmarks, including a patch which makes CLIP less susceptible to typographic attacks. Our findings demonstrate that it is possible to expand the set of tasks on which open-vocabulary models achieve high accuracy without re-training them from scratch.

## 1 Introduction

Open-vocabulary image classifiers offer the flexibility to perform any image classification task specified in natural language. Thanks to advances in large-scale pre-training, recent examples of open-vocabulary models such as CLIP and BASIC have demonstrated impressive performance on a variety of tasks without fine-tuning on task-specific data (i.e., in a zero-shot setting) [48, 27, 47, 1, 70]. For instance, the largest zero-shot CLIP model from Radford et al. [48] matches the ImageNet accuracy of a ResNet-50 trained on 1.2 million ImageNet images [14, 22].

Nevertheless, current open-vocabulary models still face challenges. The same CLIP model that matches a ResNet-50 on ImageNet has lower MNIST accuracy than simple logistic regression in pixel space. Moreover, even when zero-shot models achieve good performance, they are usually still outperformed by models trained or fine-tuned on specific downstream tasks.

A common practice to resolve such issues is to fine-tune the zero-shot model on the task of interest. However, fine-tuned models suffer from catastrophic forgetting [40, 61, 19, 29], performing poorly on tasks where the zero-shot model performed well [2, 65, 47]. Additionally, fine-tuning typically uses a task-specific classification head, sacrificing the text-based API that makes open-vocabulary models so appealing. Whereas an open-vocabulary model can perform any classification task in a zero-shot fashion, a fine-tuned model with a task-specific head can only process the specific task that it was fine-tuned on. This specialization can prevent knowledge obtained by fine-tuning on one task from transferring to other related tasks with different classes.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

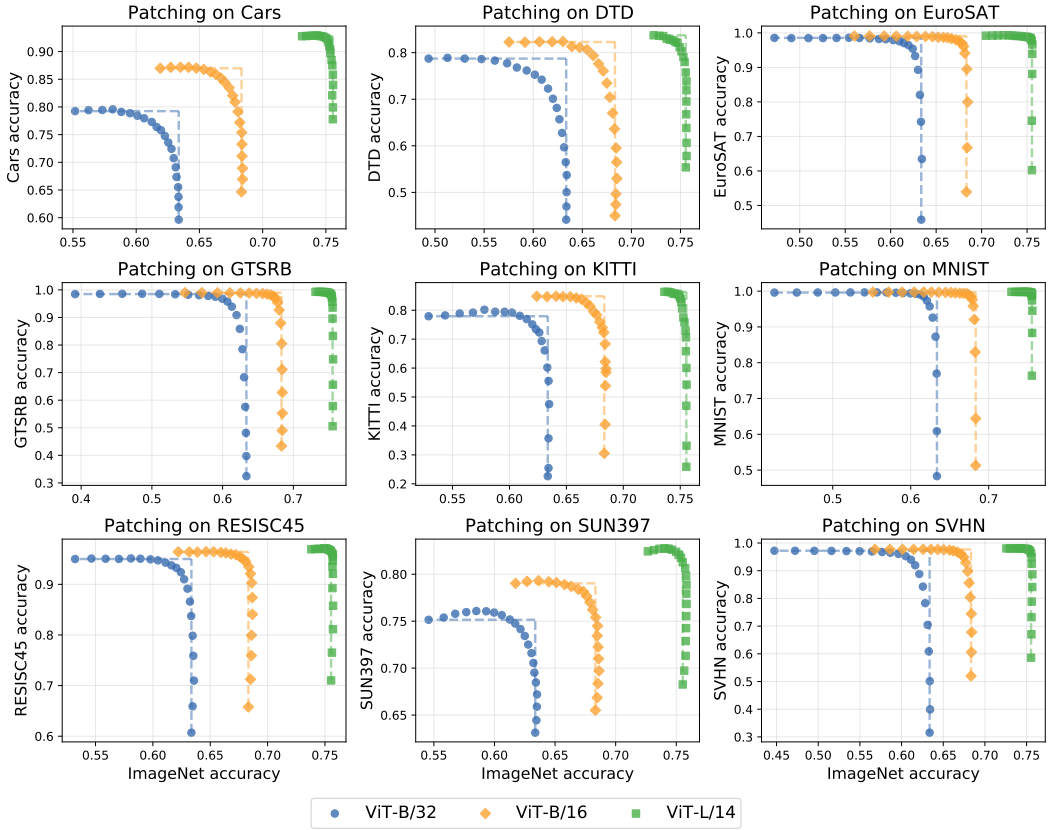


Figure 1: **Patching open-vocabulary models by linearly interpolating weights.** Given a task of interest, our aim is to improve accuracy without degrading performance on supported tasks, where accuracy is already adequate. When interpolating weights of unpatched and fine-tuned models, there is often an intermediate solution where accuracy improves on the task of interest without reducing accuracy on a representative supported task, ImageNet. Dashed lines show vertical movement from the unpatched model and horizontal movement from the fine-tuned model.

A less common alternative would be to add data from the downstream task to the pre-training dataset and train a new open-vocabulary model from scratch. The resulting model could still perform any classification task, and zero-shot performance may improve on related tasks. However, training large image-text models from scratch can require hundreds of thousands of GPU hours [48, 47, 70], which makes this approach practically infeasible in most settings.

In this paper, we study *patching* open-vocabulary models at the task level, where the goal is to increase accuracy on new target tasks without reducing accuracy on other tasks. Patching aims to combine the benefits of fine-tuning and re-training from scratch: improved performance on the task of interest, maintaining the open vocabulary flexibility, transfer between tasks, and fast adaptation time. Motivated by these goals, we extend existing fine-tuning techniques [65] to open-vocabulary settings where the class space changes. In particular, we show that a simple two-step procedure is effective for patching models: first, fine-tune on the task without introducing any task-specific parameters; then, linearly interpolate between the weights of the model before fine-tuning and the weights after fine-tuning. Linearly interpolating neural network weights [18, 45] has been previously used to improve accuracy on a single task [24, 66] or robustness to distribution shift [65].

With this procedure, accuracy can substantially improve on new tasks without degrading accuracy on unrelated tasks, as illustrated in Figure 1. For instance, patching a CLIP ViT-L/14 [48] independently on nine image classification tasks [31, 11, 23, 60, 20, 34, 7, 68, 44] improves accuracy by 15 to 60 percentage points compared to the unpatched model, while accuracy on ImageNet [14] decreases by less than one percentage point. We also observe a promising trend: patching becomes more effective with model scale (Section 4.1). When patching on nine image classification tasks, a single larger

CLIP model is competitive with using one specialized model for each task—the average accuracy difference is less than 0.5 percentage points (Section 5).

Moreover, our patching procedure enables *broad transfer* (Section 6): accuracy on related tasks can increase, even when the class space changes. For instance, we partition EuroSAT [23], a satellite image dataset, into two halves with disjoint labels. Patching a ViT-L/14 model on the first half improves accuracy on the second half by 7.3 percentage points, even though the classes are unseen.

Finally, we investigate model patching on case studies including typographic attacks [21], counting [28], and visual question answering [4] (Section 7). For instance, patching using synthetic typographic attacks leads to a model that is less susceptible to typographic attacks in the real world, improving its accuracy by 41 percentage points.

## 2 Method

This section details our method for patching models on a single and multiple tasks.

**Patching on a single task.** Given an open-vocabulary model with weights $\theta_0$ and a patching task $\mathcal{D}_{\text{patch}}$, our goal is to produce a new model $\theta_{\text{patch}}$ which achieves high accuracy on $\mathcal{D}_{\text{patch}}$ without decreasing model performance on tasks where accuracy is already acceptable. We let $\mathcal{D}_{\text{supp}}$ denote a representative supported task where model performance is adequate, and later show that the method is stable under different choices of $\mathcal{D}_{\text{supp}}$ (Section 4.2). The two-step procedure we explore for producing $\theta_{\text{patch}}$ is given below.

**Patching procedure.**
**Step 1.** Fine-tune $\theta_0$ on training data from $\mathcal{D}_{\text{patch}}$ to produce a model with weights $\theta_1$.
**Step 2.** For mixing coefficient $\alpha \in [0, 1]$, linearly interpolate between $\theta_0$ and $\theta_1$ to produce $\theta_{\text{patch}} = (1 - \alpha) \cdot \theta_0 + \alpha \cdot \theta_1$. The mixing coefficient is determined via held-out validation sets for $\mathcal{D}_{\text{supp}}$ and $\mathcal{D}_{\text{patch}}$. We refer to the resulting model as $\theta_{\text{patch}}$.

**Patching on a multiple tasks.** In practice, we often want to improve model accuracy on multiple patching tasks $\mathcal{D}^{(1)}_{\text{patch}}, ..., \mathcal{D}^{(k)}_{\text{patch}}$, which can be accomplished with straightforward modifications to the procedure above. We explore three alternatives and examine their relative trade-offs in Section 5. The three alternatives are: i) *Joint patching*, where we merge the patching tasks into a single task $\mathcal{D}_{\text{patch}} = \cup_{i=1}^{k} \mathcal{D}^{(i)}_{\text{patch}}$ before running the patching procedure; ii) *Sequential patching*, where we iteratively repeat the patching procedure above on each new task $\mathcal{D}^{(i)}_{\text{patch}}$ and let $\theta_0 = \theta_{\text{patch}}$ after each completed iteration; iii) *Parallel patching*, where we apply the first step on each task in parallel to produce fine-tuned models with weights $\theta^{(1)}_1, ..., \theta^{(k)}_1$. Then, we search for mixing coefficients $\alpha_i$ to produce $\theta_{\text{patch}} = (1 - \sum_{i=1}^{k} \alpha_i) \cdot \theta_0 + \sum_{i=1}^{k} \alpha_i \cdot \theta^{(i)}_1$. For joint and parallel patching we assume access to held-out validation sets for all tasks, while in sequential patching we only assume access to held-out validation sets from the tasks seen so far. Unless otherwise mentioned we pick the mixing coefficient $\alpha$, which optimizes average accuracy on the held-out validation sets from the supported and patching task.

## 3 Experimental setup

This section details the primary tasks and models considered, in addition to the fine-tuning and evaluation protocols.

**Tasks.** We consider a diverse set of image classification tasks from Radford et al. [48], including Stanford Cars [31], CIFAR10 [32], CIFAR100 [32], Describable Textures (DTD) [11], EuroSAT [23], Food101 [5], ImageNet [14], German Traffic Sign Recognition Benchmark (GTSRB) [60], KITTI distance [20], MNIST [34], RESISC45 [7], STL10 [12], SUN397 [68] and SVHN [44]. In most experiments, we use ImageNet as a representative supported task, although we explore other supported tasks in Section 4.2. The categorization into patching tasks or supported tasks is based on the difference in accuracy between the zero-shot and specialized models—tasks where linear probes from Radford et al. [48] outperform the zero-shot model by over 10 percentage points are categorized as patching tasks. We investigate additional patching tasks as case studies in Section 7, and provide further details in Appendix A.

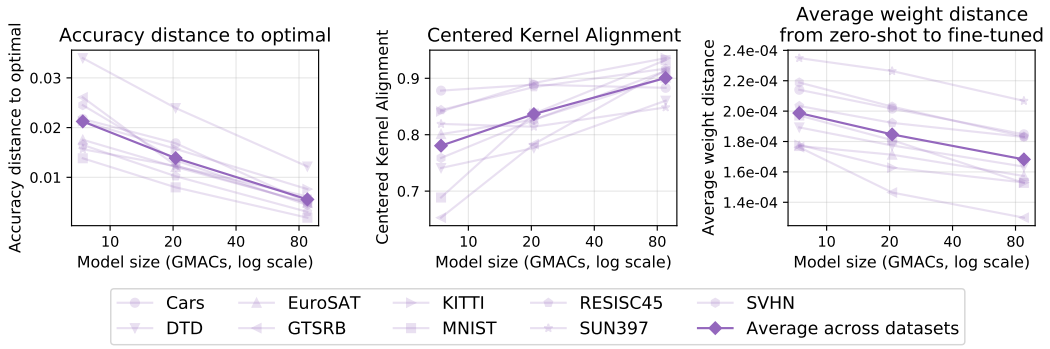


Figure 2: **Larger models are easier to patch** (left). For larger models, the unpatched and fine-tuned model are more similar with respect to their representations (center) and weights (right).

**Models.** We primarily use CLIP [48] pre-trained vision transformer (ViT) models [15]. Unless otherwise mentioned our experiments are with the ViT-L/14 model, while Section 4.2 studies ResNets [22].

**Fine-tuning on patching tasks.** Unless otherwise mentioned, we fine-tune with a batch size of 128 for 2000 iterations using learning rate 1e-5 with 200 warm-up steps with a cosine annealing learning rate schedule and the AdamW optimizer [36, 46] (weight decay 0.1). When fine-tuning, we use the frozen final classification layer output by CLIP's text tower so that we do not introduce additional learnable parameters. This design decision keeps the model open-vocabulary and does not harm accuracy, as discussed in in Appendices B and C.

**Evaluation.** We use accuracy as an evaluation metric unless otherwise stated. We use the term *average accuracy* to refer to the average of the mean accuracy on the patching tasks and the mean accuracy on the supported tasks.[1]

# 4 Patching models on a single new task

This section presents experimental results for patching a model on a single task.

As shown in Figure 1, we interpolate the weights of the zero-shot and fine-tuned model, producing a model that achieves high accuracy on both the patching tasks and the supported task. Concretely, the $y$-axis displays accuracy on the patched task while the $x$-axis displays accuracy on a supported task, ImageNet, for different mixing coefficients $\alpha$. On the nine tasks, patching improves the accuracy of ViT-L/14 by 15 to 60 percentage points, while accuracy on ImageNet decreases by less than one percentage point. Linearly interpolating weights allows practitioners to control the accuracy trade-off on the patching and supported tasks without re-training a new model.

## 4.1 The effect of scale on patching

We consistently observe that patching is more effective for larger models. Our findings are aligned with those of Ramasesh et al. [49], who observed that larger models are less susceptible to catastrophic forgetting. This section formalizes and provides insights for these observations.

**Measuring the effectiveness of patching.** We measure the effectiveness of patching via the difference in accuracy between a single model and two specialized models that achieve optimal performance on the supported task and the patching task. We refer to this measure as *accuracy distance to optimal*. Formally, let $x_\alpha$ denote the accuracy of model $\theta_\alpha$ on the supported task and $y_\alpha$ on the patching task. Accuracy distance to optimal is given by $(\max_\alpha x_\alpha + \max_\alpha y_\alpha)/2 - \max_\alpha(x_\alpha + y_\alpha)/2$. In Figure 2 (left), we show that patching becomes more effective for larger models.

**Model similarity.** Fine-tuning modifies overparameterized models less [9], which provides insights on why larger models are easier to patch: less movement is required to fit new data. We demonstrate this by evaluating representational similarity using Centered Kernel Alignment (CKA) [30] (see Appendix D for details). As shown in Figure 2 (center), the representations of the unpatched and fine-tuned models become more similar as models grow larger, indicated by larger CKA values.

[1] In other words, $(\mathbb{E}_{\mathcal{D}_{\text{supp}}}[\mathsf{Acc}(\theta, \mathcal{D}_{\text{supp}})] + \mathbb{E}_{\mathcal{D}_{\text{patch}}}[\mathsf{Acc}(\theta, \mathcal{D}_{\text{patch}})])/2$, where $\mathsf{Acc}(\theta, \mathcal{D}_{\text{supp}})$ and $\mathsf{Acc}(\theta, \mathcal{D}_{\text{patch}})$ are accuracies on supported tasks and patching tasks, respectively.

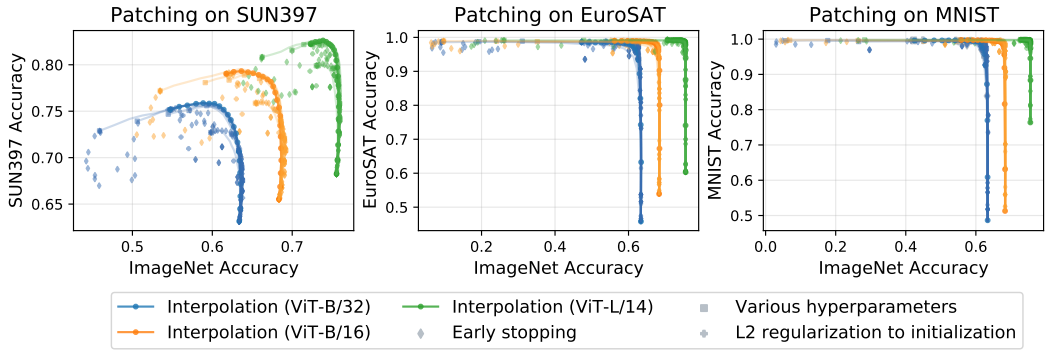


Figure 3: **The frontier of accuracy trade-offs can be recovered by linearly interpolating weights.** Interpolating the unpatched and fine-tuned models recovers the accuracy trade-off of early stopping, regularization towards the initialization, and changes in hyperparameters. Additional details and comparisons can be found in Appendix E.

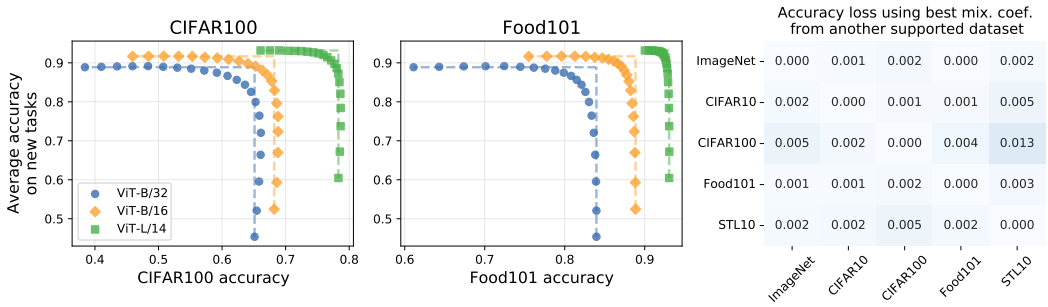


Figure 4: **Results are consistent across supported tasks.** For multiple supported tasks, we observe similar accuracy improvements on patching tasks, without substantially decreasing supported task accuracy. Additional results for the supported tasks Food101, STL10 and ImageNet are in Appendix F. Moreover, choosing the mixing coefficients using a different supported task does not substantially decrease average accuracy on patching and supported tasks (right).

Moreover, Figure 2 (right) shows that the average L1 distance between the weights of the unpatched and fine-tuned models, $||\theta_1 - \theta_0||_1/|\theta_0|$, decreases with scale.

### 4.2 Baselines and ablations

**Baselines.** There are many alternatives which enable a trade-off between accuracy on the supported and new downstream tasks. These methods include early stopping during fine-tuning, applying a regularization term which penalizes movement from initialization, or training with different hyper-parameters including a smaller learning rate. Unlike interpolation, these methods do not enable navigating the accuracy trade-off without fine-tuning the model again many times. Moreover, Figure 3 demonstrates that the accuracy trade-off frontier for early stopping, regularization, or varying hyper-parameters can be recovered by interpolating weights with different mixing coefficients. Additional baselines and discussion is provided in Appendix E.

**Additional supported tasks.** In Figure 1, we use ImageNet as a representative supported task. This section demonstrates that our procedure is stable under different choices of the supported task. Instead of ImageNet, we use CIFAR10, CIFAR100, Food101 and STL10. Figure 4 displays representative results, where performance on the nine patching tasks from Figure 1 are averaged (see Appendix F for additional results). We observe consistent results across supported tasks, and that the optimal mixing coefficients are stable across different choices of supported tasks (Figure 4, right).

**Additional models.** In addition to the CLIP ViTs used in the majority of our experiments, we study the CLIP ResNet models [22]. Results are shown in Appendix G. We find that patching is less effective for ResNets compared to ViTs of similar size, which corroborates the findings of Ramasesh et al. [49] that ResNets are generally more susceptible to catastrophic forgetting. However, similarly to ViTs, we still observe improvements with scale.

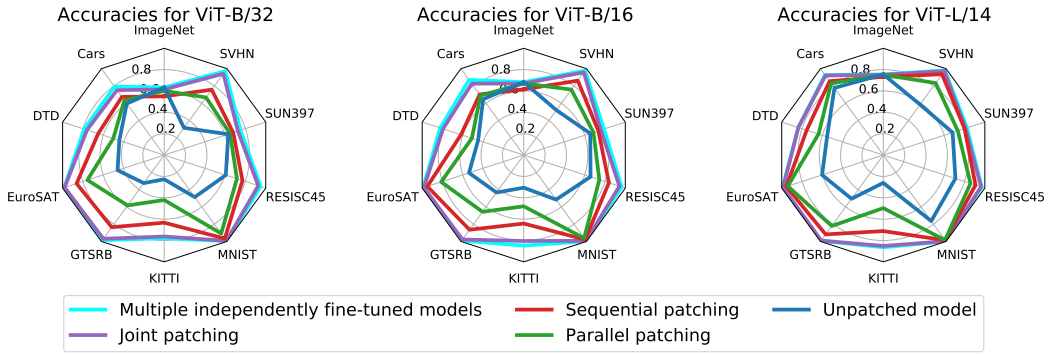


Figure 5: **Contrasting strategies for patching on multiple tasks.** ImageNet is used as the supported task while the other nine datasets are used for patching. When data from all patching tasks is available, joint patching yields a single model that is competitive with using ten different specialized models.

## 5 Patching models on multiple tasks

This section details experimental results for patching on multiple datasets. Recall from Section 2 that there are many strategies for extending our patching procedure to multiple datasets, which we briefly revisit. For *joint* patching we merge all the datasets into a single fine-tuning task and apply our patching procedure as normal. For *sequential* patching we iteratively perform our procedure once per task, using the patched model at each step as the initialization for the next step.[2] We also explore *parallel* patching, for which we have an unpatched model $\theta_0$ and independently fine-tune on each of the tasks in parallel. We then search for mixing coefficients to combine the resulting models. For tasks $1, ..., k$, let $\theta_1^{(1)}, ..., \theta_1^{(k)}$ denote the fine-tuned models for each task. The patched model is then $\theta_{\text{patch}} = (1 - \sum_{i=1}^{n} \alpha_i) \cdot \theta_0 + \sum_{i=1}^{n} \alpha_i \cdot \theta_1^{(i)}$. Since it is impractical to exhaustively search over each $\alpha_i$, we instead search over a one-dimensional scalar $\beta \in [0, 1]$ where $\alpha_i = \beta/k$ for $i \in \{1, ..., k\}$.[3] Further experimental details are provided in Appendix I.

These methods have various trade-offs and may be applicable for different scenarios. Joint patching is only possible when data from all tasks you wish to patch is available. On the other hand, sequential patching is appropriate when the tasks are observed one after another. Finally, parallel patching can leverage distributed hardware.

Figure 5 displays experimental results when patching on all nine tasks from Section 4. We observe that joint patching is the best-performing method: for ViT-L/14, joint patching achieves an average accuracy of 83.8%, a 15.8 percentage points improvement over the unpatched model, and only 0.5 percentage points lower than using multiple specialized models.[4] Moreover, patching a ViT-B/32 model with the joint strategy achieves an average accuracy 6.1 percentage points higher than a ViT-L/14 unpatched model, which requires 12x more GMACs. The accuracy of sequential patching approaches that of joint patching, especially for larger models. Additionally, parallel patching underperforms sequential patching. When patching sequentially, we found that interpolation is crucial for mitigating catastrophic forgetting, especially for smaller models. For instance, sequentially fine-tuning a ViT-B/32 without interpolation *reduces* average accuracy by 4.6 percentage points. This is compared to an average accuracy *increase* of 11 percentage points when using sequential patching. Additional results, including experiments on SplitCIFAR [51], can be found in Appendix I.

## 6 Broad transfer

An alternative to our patching approach is to introduce parameters which are specific to each new task. By contrast, the method we explore always maintains a single model. Beyond practical considerations, this section describes an interesting advantage of the single model approach: patching the model on

[2]The tasks are seen in the random order: SUN397, MNIST, DTD, GTSRB, KITTI, EuroSAT, Cars, SVHN, and RESISC45.

[3]We also explored adaptive black-box optimization algorithms to choose the mixing coefficients $\alpha_i$ [50], but observed little improvement (0.3 to 0.4 percentage points on average).

[4]Recall from Section 3 that average accuracy considers both patching and supported tasks with equal weights.

| | Cars | DTD | EuroSAT | GTSRB | KITTI | MNIST | RESISC45 | SUN397 | SVHN |
|---|---|---|---|---|---|---|---|---|---|
| Unpatched accuracy | 86.2 | 64.9 | 79.9 | 51.7 | 43.4 | 82.6 | 73.4 | 76.9 | 72.8 |
| Patched accuracy | 87.0 **(+0.8)** | 66.1 **(+1.2)** | 87.2 **(+7.3)** | 71.1 **(+19.4)** | 60.4 **(+17.0)** | 91.3 **(+8.7)** | 74.2 **(+0.8)** | 79.3 **(+2.4)** | 88.9 **(+16.1)** |

Table 1: **Patching can generalize to unseen classes.** We randomly partition each dataset into tasks $A$ and $B$ with disjoint class spaces of roughly equal size. This table reports how patching on task $A$ effects accuracy on task $B$ for the ViT-L/14 model. In all cases, accuracy on task $B$ improves when patching on task $A$ even though the classes are *unseen* during patching.

| Task $A$ | MNIST | SVHN | EuroSAT | RESISC45 | MNIST | FashionMNIST | GTSRB | MTSD |
|---|---|---|---|---|---|---|---|---|
| Task $B$ | SVHN | MNIST | RESISC45 | EuroSAT | FashionMNIST | MNIST | MTSD | GTSRB |
| Unpatched accuracy | 58.6 | 76.4 | 71.0 | 60.2 | 67.7 | 76.4 | 19.3 | 50.6 |
| Patched accuracy | 68.9 **(+10.3)** | 93.2 **(+16.8)** | 69.7 **(-1.3)** | 70.4 **(+10.2)** | 70.8 **(+3.1)** | 77.5 **(+1.1)** | 30.8 **(+11.5)** | 69.8 **(+19.2)** |

Table 2: **Patching on task $A$ can improve accuracy on a related task $B$.** For a pair of tasks $A$ and $B$, we report accuracy of the ViT-L/14 on task $B$, after patching on task $A$, finding improvements on seven out of eight cases.

task $A$ can improve accuracy on task $B$, even when task $A$ and $B$ do not share the same classes. We refer to this phenomenon as *broad transfer*. Note that we are able to study this phenomenon because the model remains open-vocabulary throughout our patching procedure.

This section explores two experiments to measure the effects on task $B$ when patching the model on task $A$. First, we explore broad transfer by randomly partitioning datasets into disjoint sets with no class overlap. For a dataset $\mathcal{D}$ we partition the class space $\mathcal{Y}$ into two disjoint sets of roughly equal size $\mathcal{Y}_A$ and $\mathcal{Y}_B$. We build task $A$ with the examples $(x, y) \in \mathcal{D}$ where $y$ belongs to $\mathcal{Y}_A$, and task $B$ with examples $(x, y)$ where $y$ belongs to $\mathcal{Y}_B$. Table 1 shows how patching a model on task $A$ affects the accuracy on task $B$ for nine datasets $\mathcal{D}$. The accuracy improvements on task $B$ range from 0.8 to 19.4 percentage points, even though the classes from task $B$ are not seen during patching.

To further understand transfer, we consider additional task pairs $A$ and $B$, which are now different datasets. While some pairs $A$, $B$ share classes, there are still instances of broad transfer. Concretely, Table 2 examines i) MNIST and SVHN, two digit recognition tasks with shared classes; ii) EuroSAT and RESISC45, two satellite imagery recognition tasks where there are unshared classes but some overlap; iii) GTSRB and MTSD [17], two traffic sign recognition datasets where there are unshared classes but some overlap; and iv) MNIST and FashionMNIST [67], which do not share any classes but appear visually similar. In seven out of eight experiments, patching on task $A$ improves accuracy by 1.1 to 19.2 percentage points on task $B$. The exception is when $A$ is EuroSAT and $B$ is RESISC45, where accuracy decreases by 1.3 percentage points.

In all experiments, when patching on task $A$ we choose the mixing coefficient $\alpha$ by optimizing the held-out validation accuracy on task $A$ and a supported task (in this experiment we use ImageNet). While it is possible for a method that introduces new parameters for each task to exhibit broad transfer to new data, this also requires knowing which parameters to apply for the new data. This is not necessary in the single model approach.

# 7 Case studies

We further examine the performance of model patching in three additional settings, which highlight weaknesses of the zero-shot CLIP model and showcase broad transfer (Section 6). We patch models so that they are i) less susceptible to typographic attacks (Section 7.1), ii) able to count objects (Section 7.2), and iii) able answer questions about images (Section 7.3).

## 7.1 Typographic attacks

Goh et al. [21] find that CLIP models are susceptible to *typographic attacks*, where text superimposed on an image leads to misclassification. For example, in Figure 6 *(a)*, the text on the pink note saying "dog" leads a CLIP to misclassify the image of a cat as a dog. To fix this vulnerability, we procedurally generate typographic attack data by adding text with incorrect class names to SUN397 [68], as seen in Figure 6 *(b)*. We then collect a test set of 110 real world images by placing notes on objects and

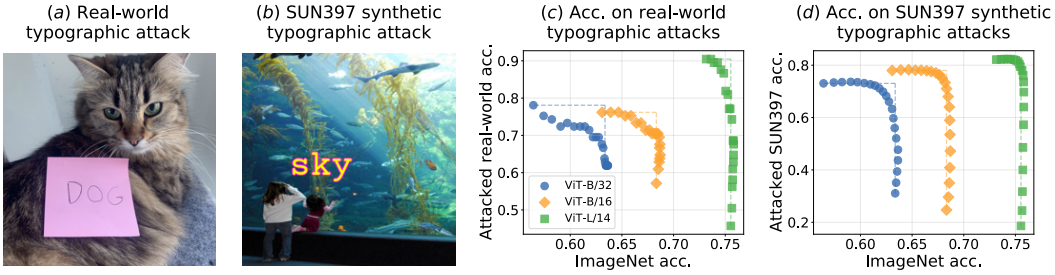


Figure 6: **Guarding against real-world typographic attacks by patching on synthetic data.** *(a)* A sample from our real-world typographic attacks test set. A CLIP ViT-L/14 is "tricked" into classifying this image as a dog instead of a cat. *(b)* Sample of synthetic typographic attack data. *(c)* Performance on real-world data with unseen classes after patching on *only* synthetic typographic attacks (curves produced by interpolating between the unpatched and fine-tuned model). *(d)* Analogous curves for the test set of the synthetic data used for patching.

taking photos. After applying the patching procedure (Section 2) on the synthetic data, we evaluate on the real-world images (Figure 6 *(c)*) and synthetic test set (Figure 6 *(d)*). We observe that while larger models are more susceptible to typographic attacks, they are also more amenable to patching. Furthermore, we see an example of broad transfer between the synthetic and real-world data: when patching ViT-L/14 on synthetic data, its accuracy on real-world typographic attacks improves 41 percentage points even though the real-world classes are unseen. The cost is a reduction of less than 1 percentage point on ImageNet. We present details on the task and data collection in Appendix J.

### 7.2 Counting

Radford et al. [48] find that CLIP models struggle to count the number of objects in CLEVR [28]. Here, the task is to choose an integer between 3 and 10 for each image, corresponding to the number of visible objects. While a straightforward way to patch such a task is to fine-tune on it directly, we investigate if patching on a subset of the classes allows the patched model to generalize to other numbers. Specifically, we patch on images with 4, 5, 6, 8, or 9 objects. To evaluate broad transfer, we test on images with 3, 7, and 10 objects (7 for understanding interpolation and 3 and 10 for extrapolation). We find that patching improves accuracy from 59% to over 99% on unseen classes with less than half a percentage point decrease in ImageNet accuracy. For more details see Appendix K.

### 7.3 Visual question answering

As shown by Shen et al. [57], zero-shot CLIP models perform poorly on visual question answering [4]. Using CLIP for VQA typically involves additional parameters—for instance, Shen et al. [57] trains a transformer [62] on CLIP features. In contrast, our procedure for patching CLIP on VQA does not introduce new parameters. Following Shen et al. [57], we contrast images with a series of text prompts, where each prompt corresponds to an option in multiple-choice VQA, formed by both the question and a candidate answer using the following template: "Question: [question text] Answer: [answer text]". We evaluate on multiple-choice VQA v1 [4], where each question is associated with 18 candidate answers. Our results, further detailed in Appendix L, show that patching is effective for visual question answering: patching ViT-L/14 improves its accuracy by 18 percentage points, while accuracy drops by less than one percentage point on ImageNet.

## 8 Related work

**Continual learning and catastrophic forgetting.** Learning tasks sequentially remains a challenge for neural networks. When a neural network learns a new task, the accuracy on other tasks often decreases, a phenomenon known as *catastrophic forgetting* [40, 61, 19, 29]. Approaches for alleviating catastrophic forgetting include: i) regularization based approaches such as EWC [29] and SI [71] which penalize the movement of parameters, ii) replay methods [51, 58, 35, 6, 54, 42], which incorporate data or gradient information from previous tasks when learning a new task, and iii) introducing task-specific parameters [55, 69, 38, 8, 63, 64]. In contrast to the aforementioned methods, patching requires no modification to the standard fine-tuning procedure. Moreover, in

contrast to regularization or replay based methods, patching requires no extra computational cost during training and in contrast to methods with task specific parameters, we maintain a single model. Having a single model is beneficial when there is new data which is similar to one of the tasks which has already been patched. Even without explicitly knowing which task the new data is similar to, we can observe accuracy improvements, which we discuss in Section 6.

Most similar to our work is Mirzadeh et al. [42], who observe high accuracy on task 0 on the linear path between a model which achieves high accuracy on task 0 and a model which is fine-tuned jointly on task 0 and 1. Moreover, they observe high accuracy on task 1 on the linear path between a model fine-tuned on task 1, and the jointly fine-tuned model. Therefore, there exists a path between a model which achieves good performance on task 0 and a model fine-tuned on task 1 along which a model can achieve high accuracy on both tasks. However, in Mirzadeh et al. [42] this combined path can be non-linear, leading them to propose a regularization and replay based method for achieving high accuracy on both tasks. In our work, we find that examining models on a linear path between $\theta_0$ (unpatched) and $\theta_1$ (fine-tuned) is often sufficient for obtaining a model which achieves high accuracy on both tasks (Figure 1). We speculate that this is due to scale and model architecture: in contrast to Mirzadeh et al. [42], we initialize with a model pre-trained on a large, heterogeneous dataset consisting of 400 million images [48], and primarily use vision transformers [15]. As shown in Section 4.2, our method performs substantially worse with ResNets [22], which are used by Mirzadeh et al. [42]. Finally, that catastrophic forgetting is less of a concern for models that are large and pre-trained has been observed by Ramasesh et al. [49], Mehta et al. [41]. Ramasesh et al. [49] also observe that vision transformers are less susceptible to forgetting than ResNets with the same number of parameters.

**Linear mode connectivity and robust fine-tuning.** Linearly interpolating neural network weights is a key step in our patching procedure. Because of nonlinear activations, it is initially surprising that linearly interpolating between two sets of weights can result in a high accuracy solution. However, a broad set of work observes that interpolating neural network weights can achieve high accuracy when part of the optimization trajectory is shared [18, 24, 45, 65, 39, 16, 66, 10]. The term linear mode connectivity was introduced by Frankle et al. [18]: two networks exhibit linearly mode connectivity if the accuracy does not decrease when using weights on the linear path between them. While Frankle et al. [18] focused on accuracy on a single task, Wortsman et al. [65] use linear mode connectivity to fine-tune models while preserving their robustness to natural distribution shifts. By interpolating the weights of a zero-shot and fine-tuned model, they find a solution which performs well both on the fine-tuning task and under distribution shift. In contrast to Wortsman et al. [65], we do not modify any task-specific parameters when fine-tuning, preserving the open-vocabulary nature of the models we patch. Unlike Wortsman et al. [65], we examine accuracy trade-offs across different tasks with potentially no class overlap. Moreover, we adapt a model to multiple tasks.

**Model interventions.** Several authors have studied the problem of updating a model to locally alter its behavior without external disruptions [59, 13, 43, 56, 53, 52]. Previous literature uses various terms to refer to this process, including model editing, patching or debugging. A popular use case is to update trained language models to reflect changes in the world (for instance, facts like who is the current president of Brazil) [26, 37, 33, 25]. Another, inspired by software engineering practice, is to "debug" language models through user interaction [53, 52], including providing corrective feedback to the models via natural language [3]. Mitchell et al. [43], De Cao et al. [13] propose training auxiliary networks to perform local edits on pre-trained models. Santurkar et al. [56] introduce a method for rewriting the prediction rules of a classifier, focusing on specific failure modes such as reliance on spurious correlations. In contrast with previous literature, our work explores patching models at the *task* level, aiming to systemically improve accuracy on a dataset—for instance, enabling a model to recognize dozens of satellite imagery classes.

# 9 Limitations and conclusion

**Limitations.** Accuracy on supported tasks can still decrease, especially for smaller models and when we patch many tasks sequentially. Moreover, our method provides no guarantees on which data the model performance might change.

**Conclusion.** We observe that it is possible to expand the set of tasks on which open-vocabulary models achieve high accuracy, without introducing new parameters, without re-training them from scratch, and without substantial catastrophic forgetting.

# References


[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

[2] Anders Andreassen, Yasaman Bahri, Behnam Neyshabur, and Rebecca Roelofs. The evolution of out-of-distribution robustness throughout fine-tuning. *arXiv preprint arXiv:2106.15831*, 2021.

[3] Anonymous. Fixing model bugs with natural language patches. URL `https://openreview.net/forum?id=blJrg3WvvDV`.

[4] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, 2015.

[5] 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.

[6] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[7] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[8] Brian Cheung, Alexander Terekhov, Yubei Chen, Pulkit Agrawal, and Bruno Olshausen. Superposition of many models into one. In *Advances in Neural Information Processing Systems*, pages 10867–10876, 2019.

[9] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32, 2019.

[10] Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. Fusing finetuned models for better pretraining. *arXiv preprint arXiv:2204.03044*, 2022.

[11] 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.

[12] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[13] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*, 2021.

[14] 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.

[15] 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. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

[16] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

[17] Christian Ertler, Jerneja Mislej, Tobias Ollmann, Lorenzo Porzi, Gerhard Neuhold, and Yubin Kuang. The mapillary traffic sign dataset for detection and classification on a global scale. In *European Conference on Computer Vision*, pages 68–84. Springer, 2020.

[18] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020.

[19] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[21] Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi: 10.23915/distill.00030. https://distill.pub/2021/multimodal-neurons.

[22] 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.

[23] 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.

[24] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[25] Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. *arXiv preprint arXiv:2110.03215*, 2021.

[26] Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. Temporalwiki: A lifelong benchmark for training and evaluating ever-evolving language models. *arXiv preprint arXiv:2204.14211*, 2022.

[27] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. *arXiv preprint arXiv:2102.05918*, 2021.

[28] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1988–1997, 2017.

[29] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[30] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[31] 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.

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

[33] Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34, 2021.

[34] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[35] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

[37] Kelvin Luu, Daniel Khashabi, Suchin Gururangan, Karishma Mandyam, and Noah A Smith. Time waits for no one! analysis and challenges of temporal misalignment. *arXiv preprint arXiv:2111.07408*, 2021.

[38] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[39] Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. *arXiv preprint arXiv:2111.09832*, 2021.

[40] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[41] Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning. *arXiv preprint arXiv:2112.09153*, 2021.

[42] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. *arXiv preprint arXiv:2010.04495*, 2020.

[43] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

[44] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[45] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*, 2020.

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[47] Hieu Pham, Zihang Dai, Golnaz Ghiasi, Hanxiao Liu, Adams Wei Yu, Minh-Thang Luong, Mingxing Tan, and Quoc V Le. Combined scaling for zero-shot transfer learning. *arXiv preprint arXiv:2111.10050*, 2021.

[48] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[49] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*, 2021.

[50] J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform. `https://GitHub.com/FacebookResearch/Nevergrad`, 2018.

[51] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[52] Marco Tulio Ribeiro and Scott Lundberg. Adaptive testing and debugging of nlp models. In *ACL 2022*, May 2022. URL https://www.microsoft.com/en-us/research/publication/adaptive-testing-and-debugging-of-nlp-models/.

[53] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*, 2020.

[54] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pages 348–358, 2019.

[55] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[56] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 23359–23373. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/c46489a2d5a9a9ecfc53b17610926ddd-Paper.pdf.

[57] Sheng Shen, Liunian Harold Li, Hao Tan, Mohit Bansal, Anna Rohrbach, Kai-Wei Chang, Zhewei Yao, and Kurt Keutzer. How much can clip benefit vision-and-language tasks? *arXiv preprint arXiv:2107.06383*, 2021.

[58] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[59] Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitriy Pyrkin, Sergei Popov, and Artem Babenko. Editable neural networks. *arXiv preprint arXiv:2004.00345*, 2020.

[60] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pages 1453–1460. IEEE, 2011.

[61] Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

[62] 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.

[63] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgwNerKvB.

[64] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

[65] Mitchell Wortsman, Gabriel Ilharco, Mike Li, Jong Wook Kim, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. *arXiv preprint arXiv:2109.01903*, 2021.

[66] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *arXiv preprint arXiv:2203.05482*, 2022.

[67] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[68] Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119 (1):3–22, 2016.

[69] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[70] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

[71] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org, 2017.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]