Supplementary Material: A Reality Check on Pre-training for Exemplar-free Class-Incremental Learning

Eva Feillet[⋆,†] eva.feillet@cea.fr
Adrian Popescu[⋆] adrian.popescu@cea.fr
Céline Hudelot[†] celine.hudelot@centralesupelec.fr

(⋆) Universite Paris-Saclay, CEA, List, F-91120, Palaiseau, France
(†) Universite Paris-Saclay, CentraleSupélec, MICS, 91190, Gif-sur-Yvette, France

A. Experimental settings

A.1. Datasets for incremental tasks

We consider four datasets containing a thousand classes each, denoted Casia, Herbarium, Inat, and Landmarks and sampled from Casia[1] [22], Herbarium 2020[2] [6], Google Landmarks v2[3] [10] and iNaturalist 2018[4] [20], respectively.

In the experiments involving generated data to augment the initial subset of classes from Casia, we use the dataset provided by the authors of DCFace[5] [7]. For the other datasets, synthetic classes of the domain are obtained by following the procedure described in subsection A.2, and ImageNet classes are selected following the method described in subsection A.3.

A.2. Generation of additional classes for pre-training

We remind that the set of classes corresponding to the initial subset of data D_1 is denoted C_1. We explain how to obtain a set of new class names $\tilde{C}_1$ from the same domain as the classes from C_1. Then, we populate visual classes by prompting a text-to-image model with the class names from $\tilde{C}_1$.

Generation of class names. We assume that each class from C_1 is described by its name c_i and by a short textual description d_i. To obtain a list of new class names from the same domain as the classes from C_1, we prompt the LLM with couples of a class name from C_1 and its description. From the output of the LLM, we form a new list of pairs $(\tilde{c}, \tilde{d})$, where $\tilde{c}$ is a new class name and $\tilde{d}$ is its associated description.

In practice, we used the pre-trained Llama-2-13b-chat[6] [19] model and the huggingface libraries `diffusers` (version 0.21.4) and `transformers` (version 4.34.1). Class descriptions are either available in the metadata or can be obtained by prompting an LLM with the following pattern: *"Here is a list of* $< domain >$ *concepts:* $\{c_1, c_2, \dots\}$. *Could you provide a short visual description of each concept?"*.

To facilitate post-processing, we first provide a system prompt to parse the output as a JSON file easily: *"role: system, content: Always answer in JSON format."* We also present class examples as JSON. *"Here is a JSON containing* $< domain >$ *names:* $\{''domain'' : \{c_1 : d_1, c_2 : d_2, \dots\}\}$. *Could you provide ten more items on the same topic, with a short visual description of each item?"*.

In our experiments, we use a maximum sequence length of 512 tokens for the input of the LLM (same for the output). To account for this constraint, we prompt the LLM multiple times with different examples and ask for only ten results each time. We randomly sample three class names from C_1 for each prompt to obtain diverse outputs. We also use different temperatures (0.6, 0.7, 0.8) and top-p values (0.9 and 0.92). We iterate the process until we obtain 1000 new unique class names.

Example with Inat. First, we associate a WordNet description with each Inat class using NLTK version 3.8.1. We take advantage of the fact that each class from iNaturalist 2018 corresponds to a living species and is provided with the species' ancestors in a natural taxonomy. We remark that each species can be associated with a WordNet [8] synset through its name or the name of an ancestor in the natural taxonomy.

Then, we use the class names along with their description to prompt the LLM: *"Here is a JSON containing fauna, flora, and fungi species with their description:* {*"species": {"name": ..., "description": ...}* }. *Could you provide 10*

[1] https://github.com/deepinsight/insightface/tree/master/recognition/_datasets_
[2] https://www.kaggle.com/c/herbarium-2020-fgvc7
[3] https://github.com/cvdfoundation/google-landmark
[4] https://github.com/visipedia/inat_comp/blob/master/2018/
[5] https://github.com/mk-minchul/dcface
[6] https://github.com/facebookresearch/llama

more species names, with a short visual description of each item?". Here are some examples of class names and descriptions provided as inputs to Llama-v2-13b-chat.

- Malurus cyaneus, largest order of birds comprising about half the known species; rooks; finches
- Microcarbo africanus, cormorants
- Dactylotum bicolor, short-horned grasshoppers; true locusts
- Heterotheca villosa, genus of yellow-flowered North American herbs
- Plantago virginica, type genus of the family Plantaginaceae; large cosmopolitan genus of mostly small.

Here are some examples of class names and descriptions provided as outputs by the LLM.

- Crocodilus porosus, saltwater crocodile; reptile
- Delphinium elatum, larkspur, a tall, slender flower with delicate petals
- Tulipa gesneriana, a type of tulip with large, showy flowers in shades of pink, yellow, and white
- Ganoderma applanatum, a species of bracket fungus with a flat, shelf-like cap and a distinctive red-brown color
- Dendrocalamus giganteus, tall bamboo grass with hollow stems and long, thin leaves.

Example with Herbarium. In the case of Herbarium, we do the same as for Inat, but only with the names of plant species.

Example with Landmarks. Using Google Landmarks v2 metadata, we assign each class to a landmark category. Then, to obtain new class names for Landmarks, we prompt the LLM as follows: *"Here is a JSON containing landmark names with their description: {"landmark": {"name": ..., "description": ...} } Could you provide 10 more landmark names, with a short visual description of each item?"*. Here are some examples of class names and descriptions provided as inputs to Llama-v2-13b-chat.

- Walls of Ávila, castle, city walls
- Hôpital Notre-Dame à la Rose, hospital
- Niagara Falls, waterfall
- Dayr-e Gachin, caravanserai
- Grote of Onze-Lieve-Vrouwekerk (Breda), church building

Here are some examples of class names and descriptions provided as outputs by the LLM.

- Kremlin wall and towers, fortress walls and towers with golden domes
- Bamboo forest, lush greenery, hiking trail, japan
- Mendenhall glacier, glacier, Alaska
- Reims cathedral, a gothic cathedral with stunning stained glass windows and ornate stone carvings,
- Kizhi pogost, a historic wooden church complex

Generation of images. For each new class from $\tilde{C}_1$, we obtain its associated images by prompting a generative model with its name $\tilde{c}$ and its visual description $\tilde{d}$. In practice, we use the Stable-Diffusion-2-1-base (SDv2.1) model, which provides high-quality images at a reasonable computational cost. We use the DPMSolverMultistepScheduler for image denoising and the CLIP text encoder to encode prompts. We use the following prompt: *"a photo of a c, d"*. We use the seeds $0, 1, 2, \ldots n-1$ to generate n images per class (202 for Herbarium, 310 for Inat, 350 for Landmarks). As reported in [17], describing the class enhances image diversity and accuracy in the case of polysemy. Following the authors of [17], we use 50 denoising steps and a guidance scale of 2.0. Other hyperparameters are used with their default values.

Example with Inat. We remark that most of the images that we generate using SDv2.1 are visually pleasant and match their class descriptions. However, the classes formed by generated images may be challenging for supervised learning in two ways: (i) some classes exhibit high visual diversity (see Figure 1), (ii) some classes are also hardly discernible from another (see Figure 2). The conjunction of high intra-class diversity and high inter-class similarity for class names, which may not be well covered by the training set of the generative model, e.g., rare plants, makes these classes less valuable for supervised pre-training.

Example with Landmarks. The generated classes are generally satisfying as they are visually pleasant and match their class description (e.g., Ephesus ruins in Figure 3). However, we remark that even well-known landmarks are subject to hallucinations; for example, the duplicated Eiffel Tower in Figure 4.

Face generation with DCFace. In the case of Casia, we augment the dataset using the recent method named Dual Condition Face Dataset Generator (DCFace) [7]. This method addresses the problem of generating a label-consistent and diverse training dataset for face recognition. It relies on a two-stage generator that controls the consistency, diversity, and subject uniqueness of the generated faces. In practice, we use the first thousand classes from the dataset released by the authors[7].

A.3. Selection of additional classes from ImageNet-21k

For each dataset, we randomly sample a thousand leaf classes from ImageNet-21k. First, we use the WordNet knowledge base to select concepts belonging to the same topic as the initial classes of a given stream, e.g. plant species for Herbarium. Then, we select the ImageNet classes that correspond to leaf concepts in WordNet and a thousand classes along those that contain at least 202 images for Herbarium, 310 images for Inat, and 350 images for Landmarks, so that the augmented set of initial classes is balanced. Note that we did not include experiments with

[7] https://github.com/mk-minchul/dcface

(a) (b) (c) (d)

Figure 1. Images generated by SDv2.1 using the prompt *"a photo of Lycopus uniflorus, a delicate, white, funnel-shaped flower with a yellow center"* and four different random seeds. Images match the class description but vary visually to an extent that is not representative of the actual species (e.g., different petal shapes in images (a) and (b)).

(a) (b) (c) (d)

Figure 2. Images generated by SDv2.1 using the prompts (a, b) *"a photo of Rhododendron ferrugineum, a flowering shrub with vibrant pink flowers"* and (c, d) *"a photo of a Rhododendron ponticum, a lowering evergreen shrub with large, showy flowers"*. Images match the description, but the two classes are visually very similar.

ImageNet classes in the case of Casia, as not enough classes matched the requirements.

A.4. Training feature extractors from scratch

In the following, we provide the hyperparameters used to train feature extractors from scratch. All our experiments are implemented with PyTorch [12]. Models are trained either on the images from the initial subset of classes D_1 containing b classes (init) or on the augmented subset $D_1 \cup \tilde{D}_1$ containing $b + x$ classes. Here b denotes the number of classes in D_1, and x denotes the number of additional classes in $\tilde{D}_1$, either synthetic (init+gen) or selected from ImageNet (init+web).

Vanilla supervised learning In our experiments with vanilla supervised learning (SL), we randomly initialize a ResNet50 network [5]. We randomly resize and crop training images to 224 by 224 pixels and flip them horizontally with a probability of 0.5. We train the model for 100 epochs with a batch size of 128, an initial learning rate of 0.1, dand ecayed after 30, 60, and 90 epochs by a factor of 0.1. We use the SGD optimizer with a momentum of 0.9. and a weight decay of 10^{-4}.

tReX In our experiments with tReX [18], we used the implementation released by the authors[8]. The architecture of the encoder is a ResNet50 network. An auxiliary module (projector) is added after the encoder during training and is discarded at test time. The projector is a means of controlling the trade-off between performance on the training task and transferability on downstream tasks. In our experiments, we use a three-layer projector because it is the version that produces the best transfer results, as reported in [18]. The rest of the hyperparameters we use also correspond to the best-transferable tReX version as per [18], i.e., one global crop with a range (0.25, 1.0), eight local crops of range (0.05,0.25), and a bottleneck dimension of 256. For experiments with the datasets Inat and Landmarks, we trained the model for 100 epochs. For experiments with Casia, we trained the model for 120 epochs because we observed a slower convergence.

In the case of Casia b100t9, we encountered difficulties making the model converge when adding 1000 synthetic classes and using the default hyperparameters of tReX. We hypothesize that the convergence problem occurs because the information contained in small crops of synthetic images of faces may not be sufficiently consistent for super-

[8] https://github.com/naver/trex

(a) (b) (c) (d)

Figure 3. Images generated by SDv2.1 using the prompt *"a photo of Ephesus, ancient city, ruins"*. Different seeds produce different viewpoints and lighting conditions, like in the images of the original Landmarks dataset.

(a) (b) (c) (d)

Figure 4. Examples of visual hallucination in images generated by SDv2.1 using the prompt *"a photo of Eiffel Tower in Paris, an iconic tower with lattice-like structure and a beautiful view of the city"*. The tower is depicted twice (a,c) or with a fence in the background (b). In this case, the class description mentioning "a lattice-like structure" may not enhance generation quality.

vised learning. So, we increased the resolution of global crops from (0.25, 1.0) to (0.9, 1.0) and from (0.05, 0.25) to (0.5, 0.9) for local crops. This enabled convergence but with relatively low accuracy compared to the other experiments with additional synthetic classes. As the authors of [7] remark, faces generated with DCFace lack 3D consistency across poses. There is still a performance gap between real (natural) and synthetic datasets for training a face recognition model.

A.5. Pre-trained baselines

In our study, we compare models trained from scratch on the initial subset of classes with models pre-trained on larger datasets.

- We used the weights from PyTorch hub[9] for the ResNet50 model trained on ImageNet-1k.
- We used the checkpoint from the authors of TinyCLIP [21] for the ResNet50 model pre-trained and distilled on LAION-400m using OpenCLIP ViT-B/32 as teacher model[10]. TinyCLIP is a recent method for distilling large-scale vision-language models. It is able to largely reduce model size while maintaining competitive zero-shot performance and linear probing accuracy for downstream tasks.
- We used the following pre-trained vision transformer (ViT) models from `timm` library[11]:
  - ViT-small trained on ImageNet-1k with AugReg,
  - ViT-small model trained on ImageNet-21k with AugReg,
  - ViT-small model trained on LVD-142m with DINOv2, and
  - ViT-small model trained on LVD-142m with DINOv2 and distilled on ImageNet-1k[12].
- We also consider "first-session adaptation" as in [11], i.e., fine-tuning a pre-trained model on the initial classes and then freezing the weights for transfer. We implemented the parameter-efficient transfer learning procedure of AdaptFormer [1]. We apply it to the ViT-S network pre-trained on ImageNet-21k and fine-tune

[9] https://pytorch.org/vision/main/_modules/torchvision/models/resnet.html#ResNet50_Weights

[10] https://github.com/wkcn/TinyCLIP

[11] https://github.com/huggingface/pytorch-image-models/blob/main/timm/models/vision_transformer.py

[12] https://github.com/facebookresearch/dinov2

it on the initial classes. We make a hyperparameter search with an initial learning rate of 0.05 or 0.01 and 10, 20, or 50 fine-tuning epochs. We use a cosine annealing learning rate scheduler with a minimum learning rate of 10^{-8}.

We have chosen these baseline models to encompass different sizes and subjects of pre-training datasets, different network architectures, and different training paradigms (supervised, self-supervised). ImageNet-1k and ImageNet-21k are commonly used for fine classification tasks and cover many concepts, such as animals, foods, plants, or objects. They provided longstanding baseline models for transfer learning. The models pre-trained on LAION-400m and LVD-142m are stronger baselines, as these datasets are richer and more diverse. We expect all pre-trained models to produce high-quality representations for the Inat classes, as natural species are well represented in the pre-training datasets. LVD-142m includes about 10% of images close to Landmarks concepts. This includes Google Landmarks v2 and an augmented version of Google Landmarks v2 obtained by web crawling, so we expect the corresponding models to transfer well to the Landmarks data stream. On the contrary, there are few human faces in the pre-training datasets, so we expect the Casia classification tasks to be a challenge for the baseline models.

A.6. EFCIL algorithms

For a given set of initial classes and a given feature extractor, we first compute the images' embeddings. For each class, a class prototype is computed by averaging the embeddings of the training samples belonging to the class. Then, we train the four EFCIL algorithms using the hyperparameters listed below. Each of the following algorithms relies on a fixed image encoder and performs inference by comparing the embedding of an input image with the mean embedding vector of the past classes. The algorithms differ in the way this comparison is implemented. Our implementation of DSLDA [13] [4], FeTrIL[14] [14] and FeCAM[15] [3] is based on the original repository of the authors.

NCM. At inference, the prediction for a given test sample is computed as the class whose prototype is the nearest to the test sample's embedding in the sense of the cosine distance.

DSLDA. This algorithm performs a linear discriminant analysis. It updates a covariance matrix and bias vector online. We do not modify the default hyperparameters. The value of the shrinkage parameter before matrix inversion is 0.0001.

FeTrIL. The method classifies images based on linear SVCs trained using `scikit-learn` implementation [13]. Hyperparameter values are $C = 1.0$ and $toler = 0.0001$.

FeCAM. For a fair comparison with the other algorithms regarding memory requirements and computing at inference time, we use the version of FeCAM with one common covariance matrix for all seen classes. We tune the shrinkage coefficients by exploring the range of values 0.0 (no shrinkage), 0.01, 0.1, 1.0, 5.0, 10.0.

A.7. Indicators.

In the following, we summarize the indicators used in our experimental study.

Accuracy. EFCIL algorithms are commonly compared based on their *average incremental accuracy*, computed as:

$$A = \frac{1}{T}\sum_{t=1}^{T} Acc_t^{[1:i]}, \tag{1}$$

where $Acc_t^{[1:i]}$ is the accuracy of the model $\mathcal{M}_t$ on test samples from $\bigcup_{j=1}^{i} D_j$, after learning at step s_t [16].

Forgetting. The *average forgetting* is computed as [9]:

$$F = \frac{1}{T-1}\sum_{i=1}^{T-1} f_i, \tag{2}$$

where the individual forgetting value is computed by:

$$f_i = \max_{i \le k \le T} Acc_k^i - Acc_T^i, \tag{3}$$

i.e., it is the difference between the best accuracy achieved for D_i at any step s_k by a model $\mathcal{M}_k$, and the accuracy of the final model $\mathcal{M}_T$ on D_i.

Ranks. Let us consider a feature extractor Φ and a data stream $\mathcal{D} = D_1 \cup D_2 \cup \ldots$. We denote by Q the matrix containing the stacked embeddings of the test samples from the first subset of classes D_1, computed using Φ. We denote by K the dimension of the feature vectors and by N the number of samples used to compute Q. In our analysis, we use the following indicators to compute the rank of the feature matrix Q.

- The number of eigenvalues to explain 80% of the variance of the feature matrix, using a PCA (computed using scikit-learn [13]).
- The number of eigenvalues to explain 80% of the variance of the covariance matrix $Cov(Q^t \cdot Q)$, using a PCA.
- The number of eigenvalues above a given threshold α [15], e.g.

$$rank(Q) = \sum_{k=1}^{min(N,K)} \mathbf{1}_{\sigma_k > \alpha}, \tag{4}$$

where σ_k is the k-th eigenvalue of Q, α is computed as $max_i \sigma_i \times max(K, N) \times \epsilon$ and ϵ is a small constant depending on the float precision, here 10^{-7}.

[13] https://github.com/tyler-hayes/Deep_SLDA
[14] https://github.com/GregoirePetit/FeTrIL
[15] https://github.com/dipamgoswami/FeCAM

- The information-based rank proposed by [2], defined as:

$$RankMe(Q) = exp(-\sum_{k=1}^{min(N,K)} p_k log(p_k)), \quad (5)$$

where p_k is computed as $\frac{\sigma_k(Q)}{||\sigma(Q)||_1} + \epsilon$. RankMe provides a smooth measure of the embeddings' rank and can be used to predict the linear probing performance of Φ, even on an unseen dataset, if the tasks of the source dataset of the encoder and of the target task are relatively close.

The intuition motivating the use of ranks to estimate the transferability of a given model for an EFCIL stream is that as long as the training dataset of the encoder is relatively diverse or near to the domain of the stream, an encoder with a greater embeddings' rank on the initial subset of classes D_1 will also have a greater embeddings' rank on the next subsets D_2, D_3... and this, in turn, correlates with higher accuracy with linear probing. However, we note that the EFCIL algorithms of our study go beyond linear probing, so we do not obtain correlations as strong as the ones reported by [2] in the case of linear probing.

B. Visualization of results

B.1. Comparing vanilla supervised learning (SL) and tReX training methods

In Figure 5 we show the average incremental accuracy of EFCIL algorithms that use a feature extractor trained on the initial subset of classes either with SL or with tReX, as a function of the initial accuracy. Experiments that correspond to a higher initial accuracy and to a higher average incremental accuracy correspond to models with an increased performance on the initial subset of data and an increased transferability on the next subsets of classes. We observe that DSLDA, FeTrIL, and tReX generally benefit from using a feature extractor trained with tReX. On the contrary, NCM performance drastically deteriorates both in terms of initial accuracy and average incremental accuracy when using tReX instead of SL.

This trend is also visible in Figure 6, where the average forgetting of the EFCIL algorithms is reported as a function of their initial accuracy. Except for NCM, the EFCIL algorithms used in our experiments exhibit a lower forgetting when using tReX.

We measured the cosine distance between a test sample and its prototype for models trained with SL and with tReX. On average, this distance is reduced when using tReX in comparison with SL (i.e. points are tighter packed). We also measure that the ratio of the cosine distances between (i) a test sample and its true prototype and (ii) the same test sample and its nearest confounding prototype is reduced. This is coherent with the lower performance of the NCM classifier when using tReX. However, classifiers that model more complex distributions are able to benefit from the representation obtained with tReX.

For each dataset, we illustrate with UMAP projections the distribution of seven classes in the latent space of feature extractors trained on the first one hundred classes of the dataset using either SL or tReX, without additional classes in Figure 7, or with a thousand additional synthetic classes in Figure 8. We remark that the shape of the clusters is quite different from one method to another and slightly changes when adding a thousand synthetic classes to the training set of the feature extractor, especially in the case of Casia.

We provide below the list of class names and descriptions for each dataset of Figure 7 and Figure 8.

Landmarks:

0. Haleakala National Park (parks, National Park of the United States)
1. Sofiyivsky Park, (parks, botanical garden in Ukraine)
2. Purana Qila (castle / fort, fortification in India)
3. Lake Como (lake, lake in Italy)
4. Çufut Qale (castle / fort, ancient fortress in Ukraine)
5. Lok Virsa Museum, Islamabad (museum, art museum in Pakistan)
6. Lake Bled (lake, lake in Slovenia).

Inat:

0. Microcarbo africanus, (Animalia, Chordata, Aves, Suliformes, Phalacrocoracidae, Microcarbo, cormorants)
1. Aphylla angustifolia (Animalia, Arthropoda, Insecta, Odonata, Gomphidae, Aphylla, dragonflies and damselflies)
2. Eryngium planum (Plantae, Tracheophyta, Magnoliopsida, Apiales, Apiaceae, Eryngium, large genus of decorative plants with thistlelike flower heads; cosmopolitan in distribution)
3. Helophilus pendulus (Animalia, Arthropoda, Insecta, Diptera, Syrphidae, Helophilus, a large order of insects having a single pair of wings and sucking or piercing mouths; includes true flies and mosquitoes and gnats and crane flies)
4. Buteogallus urubitinga (Animalia, Chordata, Aves, Accipitriformes, Accipitridae, Buteogallus, hawks; Old World vultures; kites; harriers; eagles).
5. Camassia quamash (Plantae, Tracheophyta, Liliopsida, Asparagales, Asparagaceae, Camassia, genus of scapose herbs of North and South America having large edible bulbs).
6. Chlidonias niger (Animalia, Chordata, Aves, Charadriiformes, Laridae, Chlidonias, gull family: gulls and terns)

Casia

0. 0781981: male, middle-aged

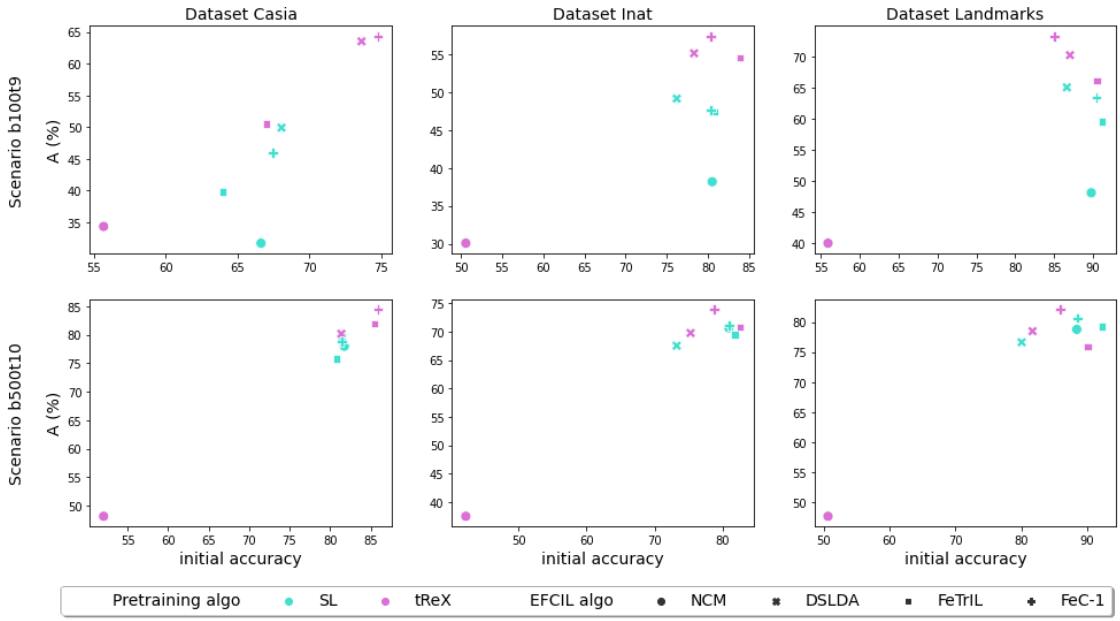

Figure 5. Comparing SL and tReX from the point of view of their initial accuracy and average incremental accuracy (A). *Values in the upper right corner are better.*

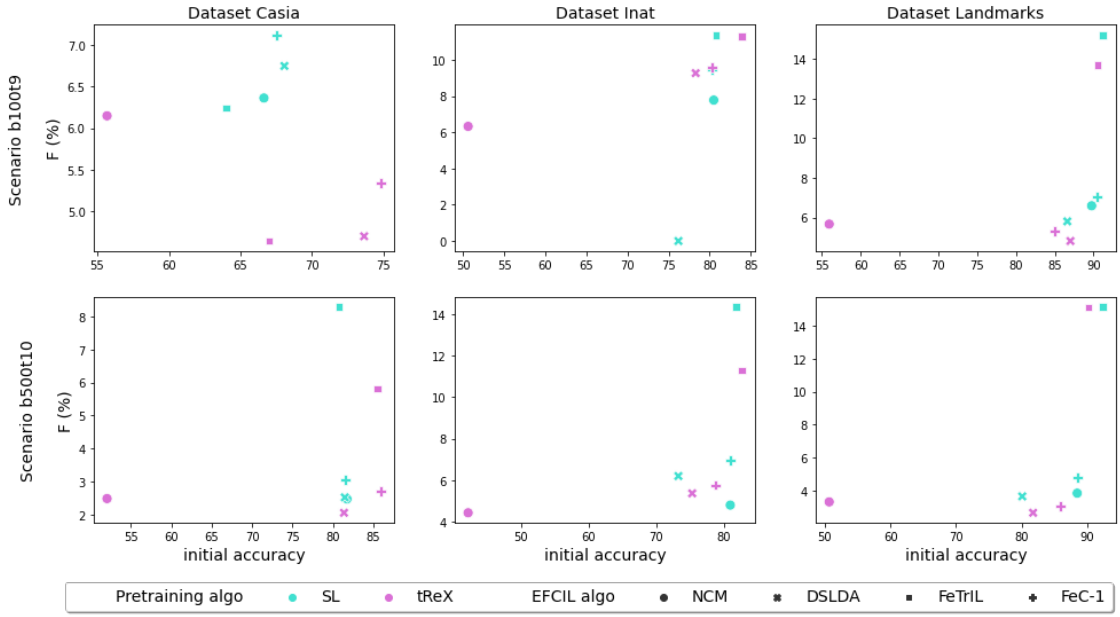

Figure 6. Comparing SL and tReX from the point of view of their initial accuracy and average forgetting (F). *Values in the lower right corner are better.*

1. 0278304: female, young
2. 0004980: female, middle-aged
3. 0000610: male, young
4. 0000389: male, young
5. 3592338: female, young
6. 0001099: male, young

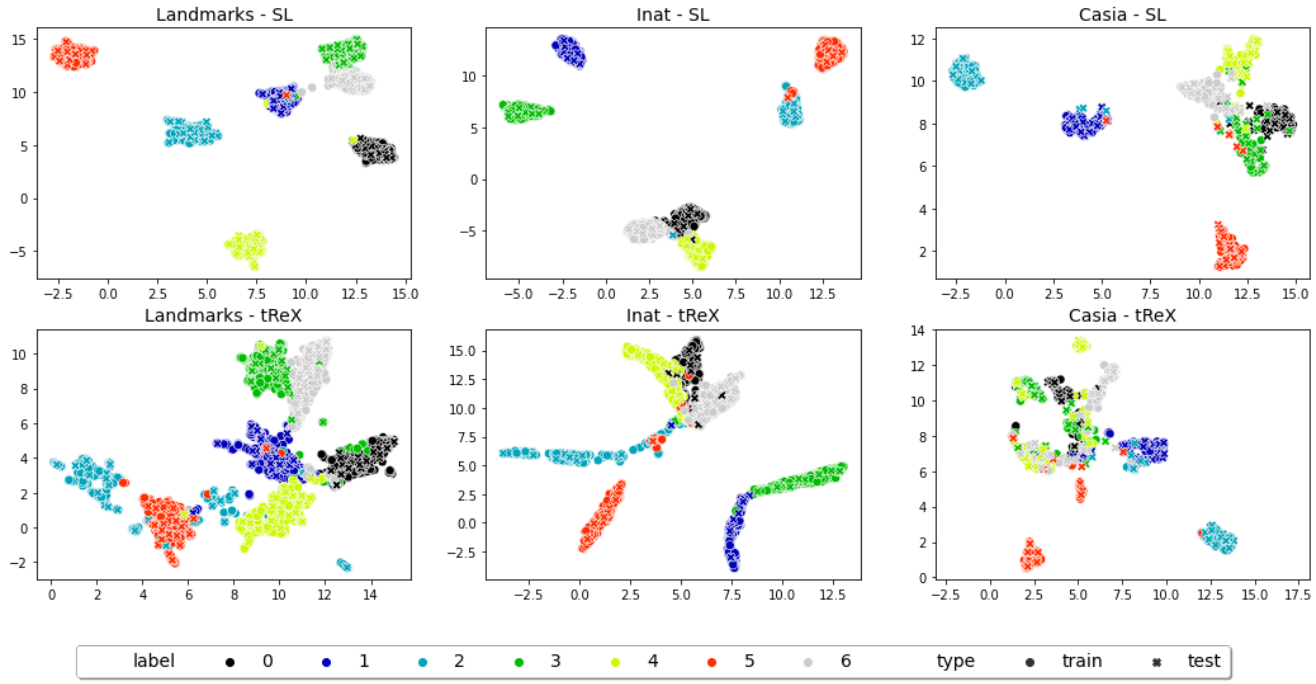

Figure 7. UMAP projection of seven classes for various feature extractors trained on the first 100 classes of the data stream, either using vanilla supervised learning (SL) or tReX.

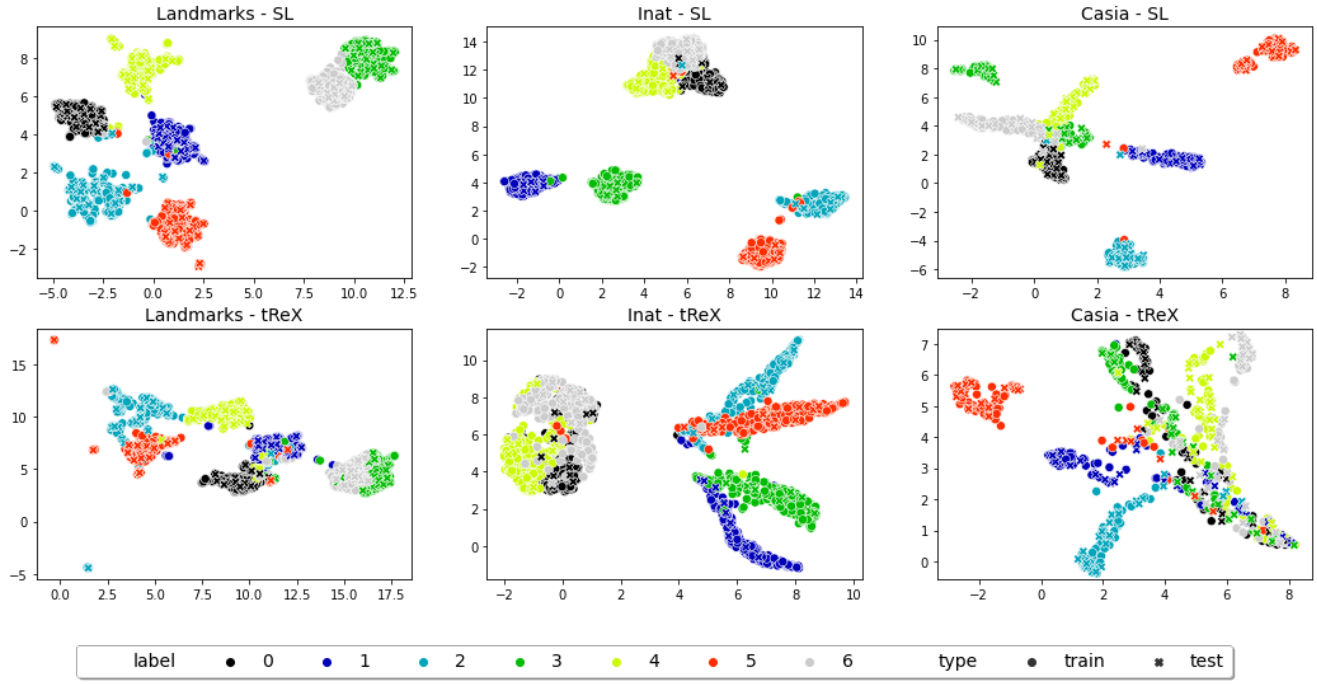

Figure 8. UMAP projection of seven classes for various feature extractors trained on the first 100 classes of the data stream augmented by a thousand synthetic classes, either using vanilla supervised learning (SL) or tReX.

B.2. Comparing baseline feature extractors

In Figure 9, we report the average incremental accuracy of each of the four EFCIL algorithms used in our study, in combination with each of the six pre-trained models used as baseline for feature extraction. We observe that the ResNet50 model and the ViT-small model trained on ImageNet-1k are consistently the least-performing feature extractors. ResNet50 pre-trained on LAION-400m comes close to the ViT-S pre-trained on LVD-142m in the experiments with Casia but has difficulties with the fine-grained natural concepts of Inat. On the contrary, the ViT-small model trained on ImageNet-21k with AugReg comes close to the Vit-small models pre-trained on LVD-142m (original version, or distilled version using ImageNet-1k), because ImageNet-21k entails many natural concepts too. The Landmarks data streams are best classified when using feature extractors trained on the largest databases LAION-400m and LVD-142m, which cover the topic of scenes and places more extensively than ImageNet-1k and ImagNet-21k do.

B.3. Comparing the efficacy of scratch training versus large-scale pre-training

In Figure 10 and Figure 11, we show the average incremental accuracy of each EFCIL algorithm as a function of the number of samples used to train the feature extractors. The results highlight the prevalent role of the domain shift between the initial training set and the incremental tasks on EFCIL accuracy. The results reported for FeCAM in Section 4 hold for the other algorithms NCM, DSLDA, and FeTrIL. For Casia, we observe that the baseline models are ineffective. This is coherent with the fact that none of the external pre-training datasets specifically cover human faces. ImageNet-21k and LVD-142m, primarily built on top of ImageNet-21k, include many natural species, and we observe that the accuracy of the associated pre-trained models is high for Inat. LVD-142m also includes an extended version of Google Landmarks v2, from which the Landmarks dataset used in our experiments is sampled. Despite this intersection, the models trained on the augmented initial subset outperform those trained on the LVD-142m dataset, which is a thousand times larger.

B.4. Choosing a feature extractor from the initial dataset

In Figure 12, we reproduce Figure 7 in higher resolution. It compares linear regressions computed on the results of EFCIL experiments with pre-trained baselines (black line) and EFCIL experiments with feature extractors trained on the initial classes (red line).

To complement Figure 12, we report the Pearson correlation coefficient between various indicators:

- B: number of classes in the initial subset of classes
- X: number of synthetic classes added in the training set of the feature extractor
- n_samples_all, n_samples_init, n_samples_incr: the number of training samples in the whole data stream $\mathcal{D} = \bigcup_{i=1}^{T} D_i$, in the initial subset of classes D_1 and in the incremental steps $D_2 \cup D_3 \dots D_T$, respectively
- for each EFCIL algorithm (NCM, DSLDA, FeTrIL, FeCAM):
  - avg_acc_incr: average incremental accuracy,
  - init_acc: initial accuracy Acc_1^1,
  - last_acc: final accuracy $Acc_T^{[1:T]}$,
  - avg_f: average forgetting F,
  - init_f: forgetting on the initial subset of classes only f_1 (see Equation 3),
  - incr_f: average forgetting on the incremental tasks only $\frac{1}{T-2}\sum_{i=2}^{T-1} f_i$.
- ranking indicators for the initial subset of data (init), for the subsequent subsets of data (incr), or for the whole dataset (all), with each of the four methods presented in appendix A.7:
  - norm_n80: number of principal components to account for 80% of explained variance of the feature matrix
  - norm_r: rank computed as per Equation 4
  - norm_info_r: rank computed as per Equation 5
  - norm_n80_cov: number of principal components to account for 80% of explained variance of the covariance matrix computed from the feature matrix.

Ranks are normalized by the total number of dimensions of the embedding space to account for the different dimensions of latent spaces across architectures, e.g. 2048 dimensions for ResNet50 versus 384 for ViT-small. A white row or column corresponds to constant values (e.g. the threshold is too low for the basic rank indicator in the case of SL).

We remark that different correlation patterns appear in Figure 13 (SL), Figure 14 (tReX), and Figure 15 (pre-trained baseline models). Correlations between the accuracy of the EFCIL methods are very high in the case of pre-trained baselines and of EFCIL experiments trained with SL. They are weaker in the case of tReX, indicating that different EFCIL algorithms do not benefit alike from the embeddings produced by tReX (e.g., NCM versus FeCAM).

We remark that EFCIL experiments that use a feature extractor trained with SL (Figure 13) exhibit a relatively

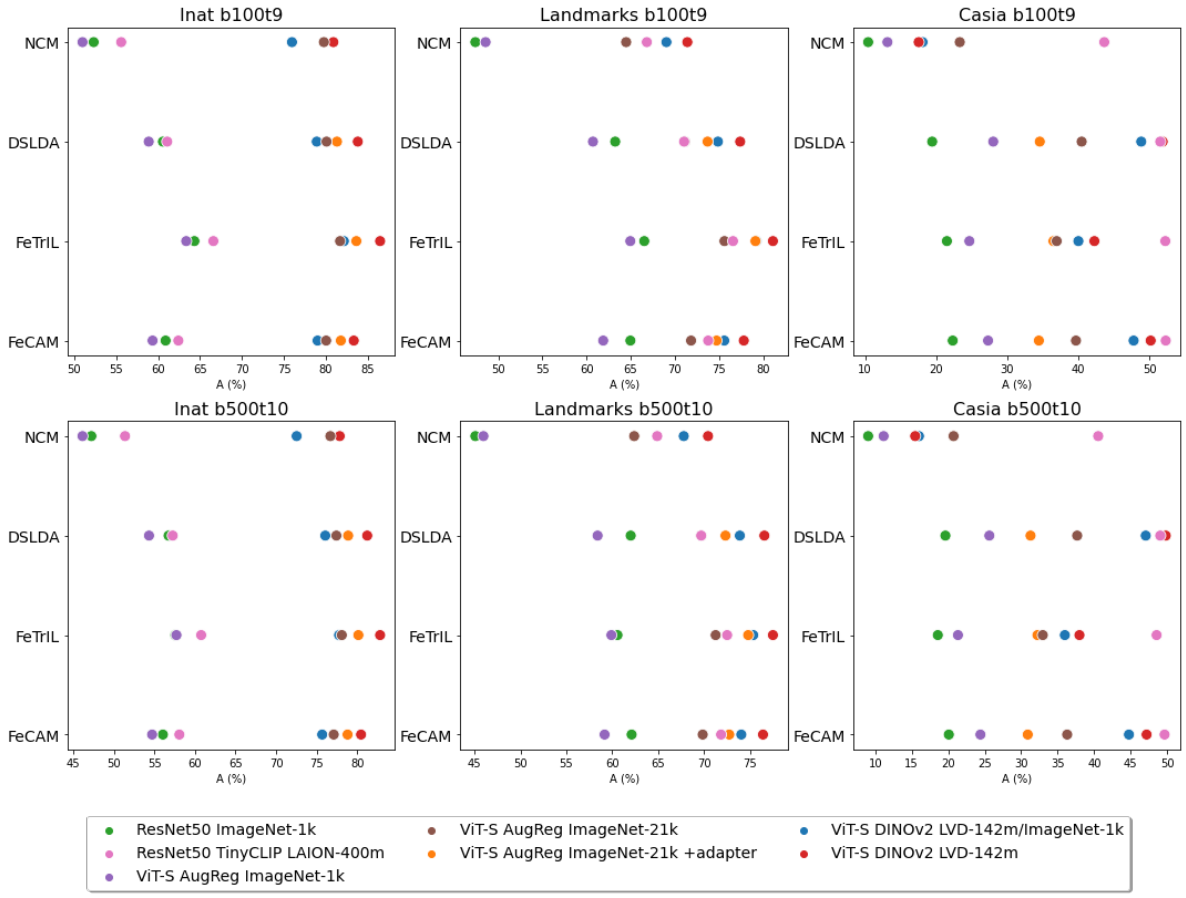

Figure 9. Comparison of pre-trained baseline models. Average incremental accuracy obtained with four EFCIL algorithms (NCM, DSLDA, FeTrIL, FeCAM).

strong correlation (around 0.7) between their average incremental accuracy and the rank of the initial feature matrix. This could offer the basis for a heuristic to choose a feature extractor at the beginning of the incremental learning process. A positive correlation is obtained for the other ranking methods, e.g., the information-based rank of the initial feature matrix, but is less strong (0.40 to 0.66). Accuracy and rank are mildly correlated in the case of experiments with tReX (Figure 14). In the case of experiments with pre-trained baselines (Figure 15), the average incremental accuracy of EFCIL algorithms is most correlated with the rank of the feature covariance matrix.

We note that when combined with a pre-trained baseline model or with a feature extractor trained with tReX, FeTrIL's forgetting tends to increase with its accuracy, which is not desirable behavior. FeTrIL is based on the assumption that classes whose prototype is close in latent space are similarly distributed. This does not seem to be the case in these experiments. The average incremental accuracy of FeTrIL is nevertheless positively correlated with the initial accuracy.

Finally, we note that some of the issues discussed previously can also be illustrated from the correlation matrices, e.g. training a model with SL on more data (real or synthetic) improves the performance of the EFCIL algorithms (Figure 13), whereas there are exceptions with tReX, e.g. the non-parametric classifier NCM behaves differently to the other EFCIL algorithms and does not benefit from tReX (Figure 14).

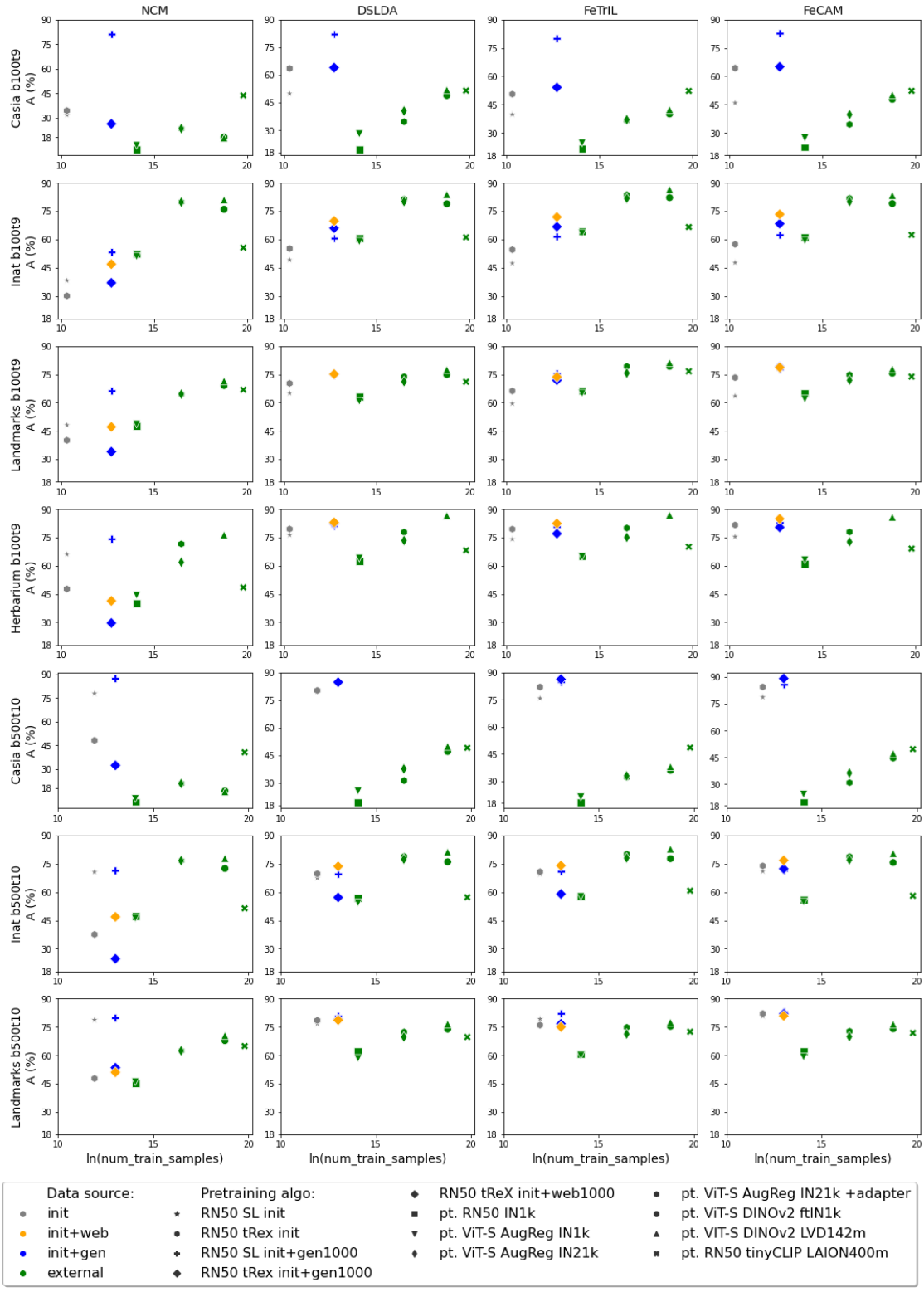

Figure 10. Comparison of pre-training methods using NCM, DSLDA, FeTrIL, or FeCAM for b100t9 and b500t10 scenarios. For each experiment, we plot the average incremental accuracy as a function of the log number of samples used to train the feature extractor.

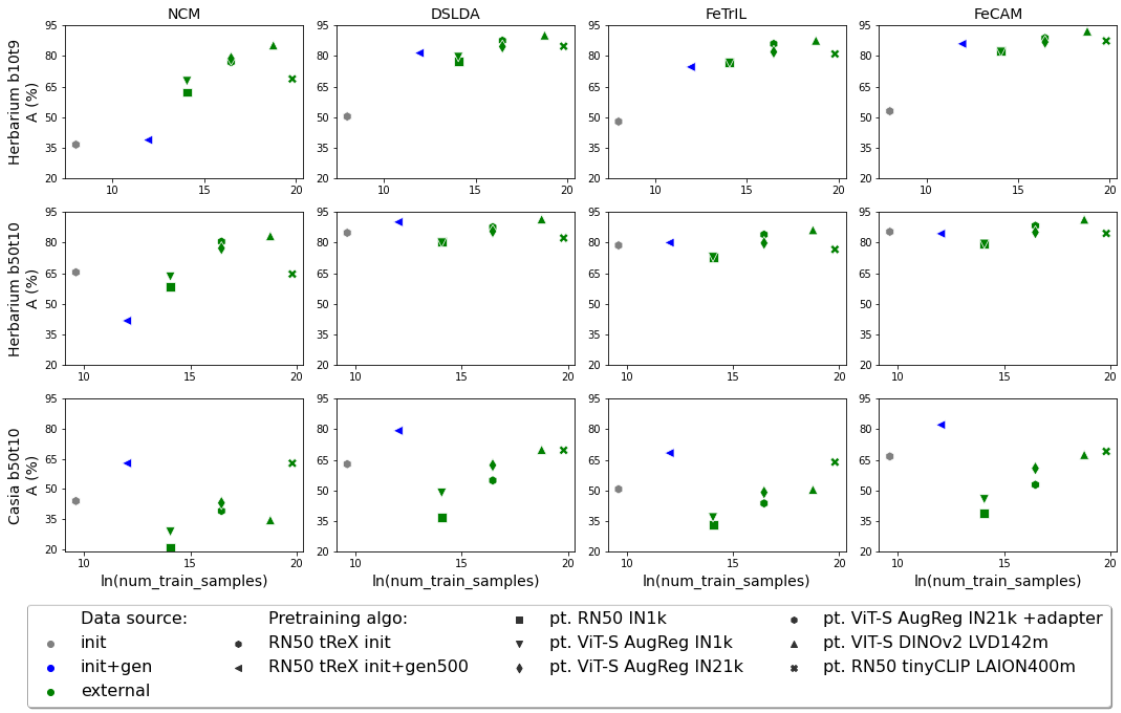

Figure 11. Comparison of pre-training methods using NCM, DSLDA, FeTrIL, or FeCAM for b10t9 and b50t10 scenarios. For each experiment, we plot the average incremental accuracy as a function of the log number of samples used to train the feature extractor.

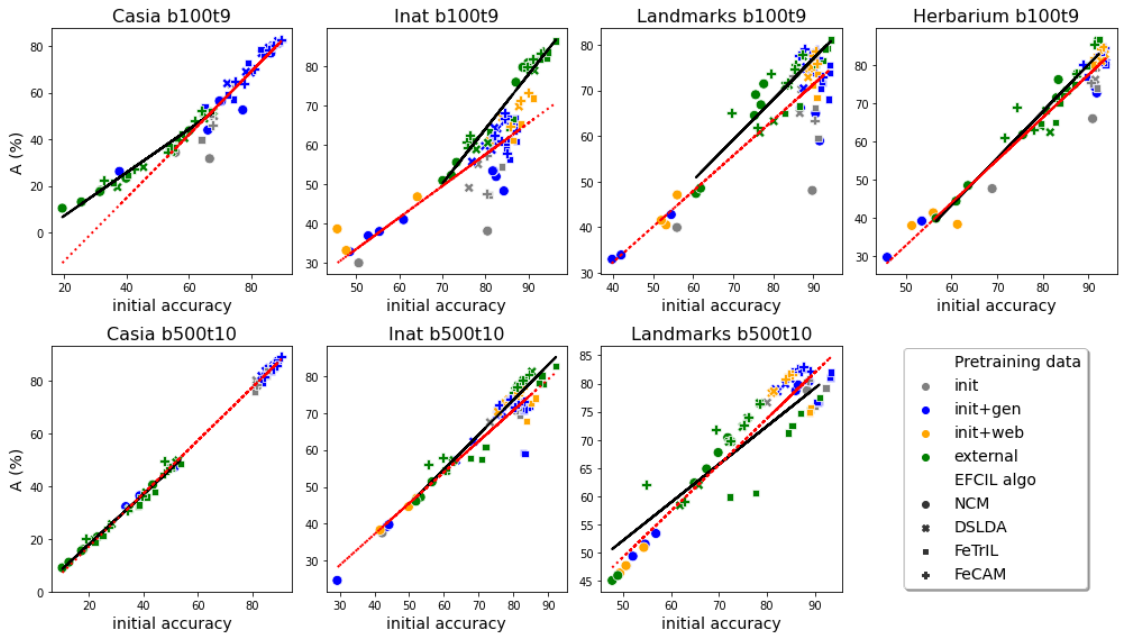

Figure 12. Average incremental accuracy as a function of the initial accuracy. Linear regressions fitted on the average incremental accuracies of four EFCIL algorithms obtained with models pre-trained on large external datasets (full, black line) or the first subset of classes with or without synthetic augmentation (dashed, red line).

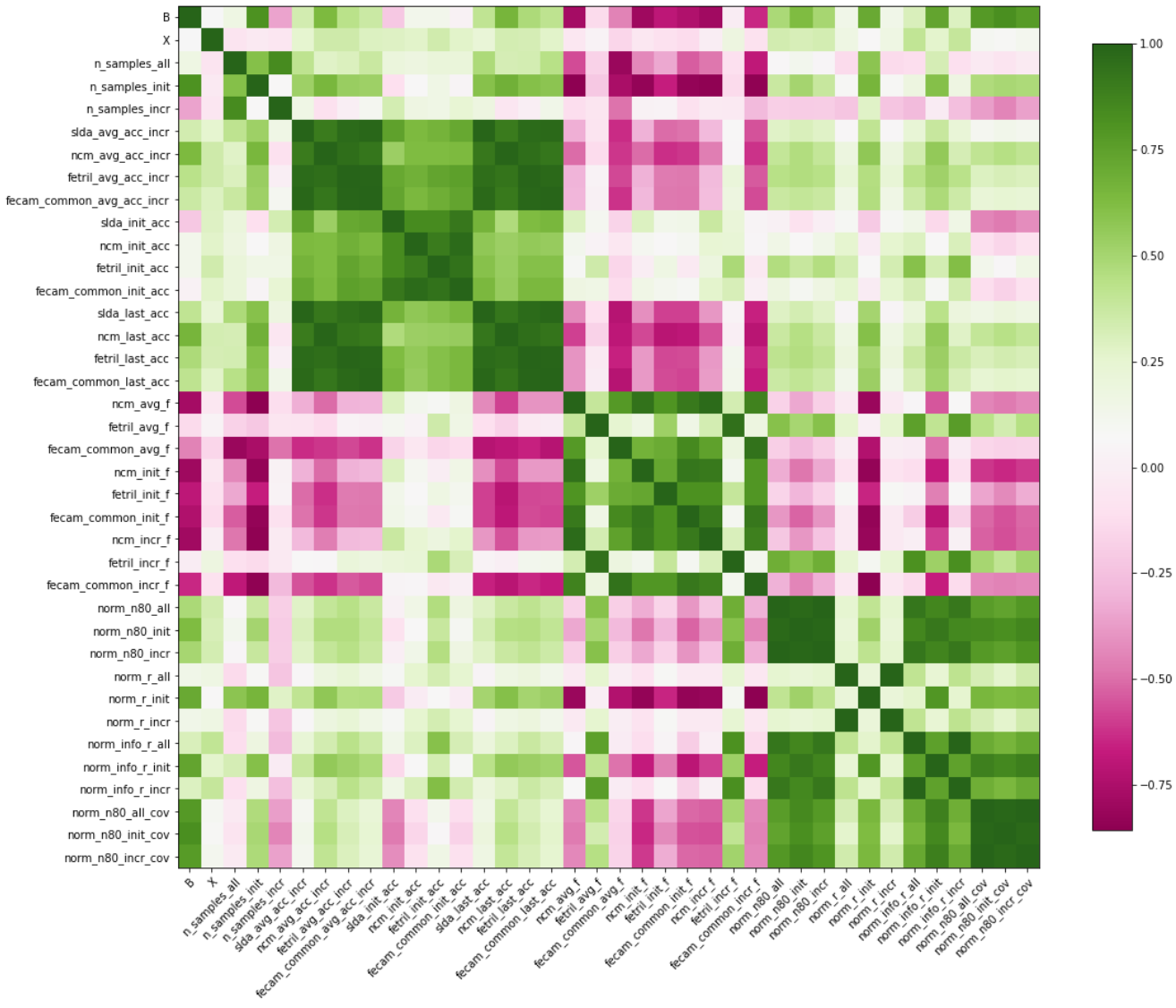

Figure 13. Pearson correlation coefficients for EFCIL experiments using vanilla supervised learning to train the feature extractor.

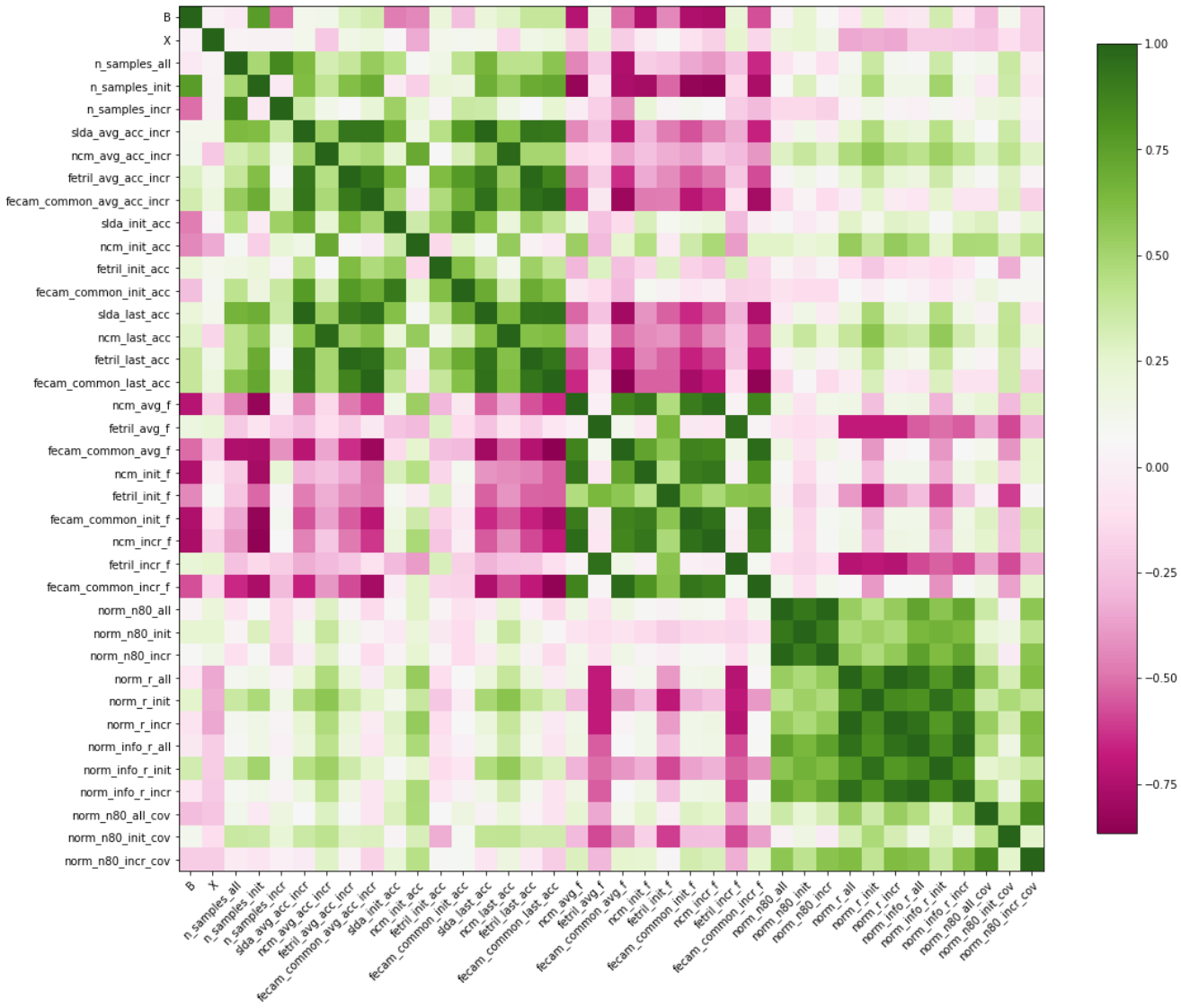

Figure 14. Pearson correlation coefficients for EFCIL experiments using tReX to train the feature extractor.

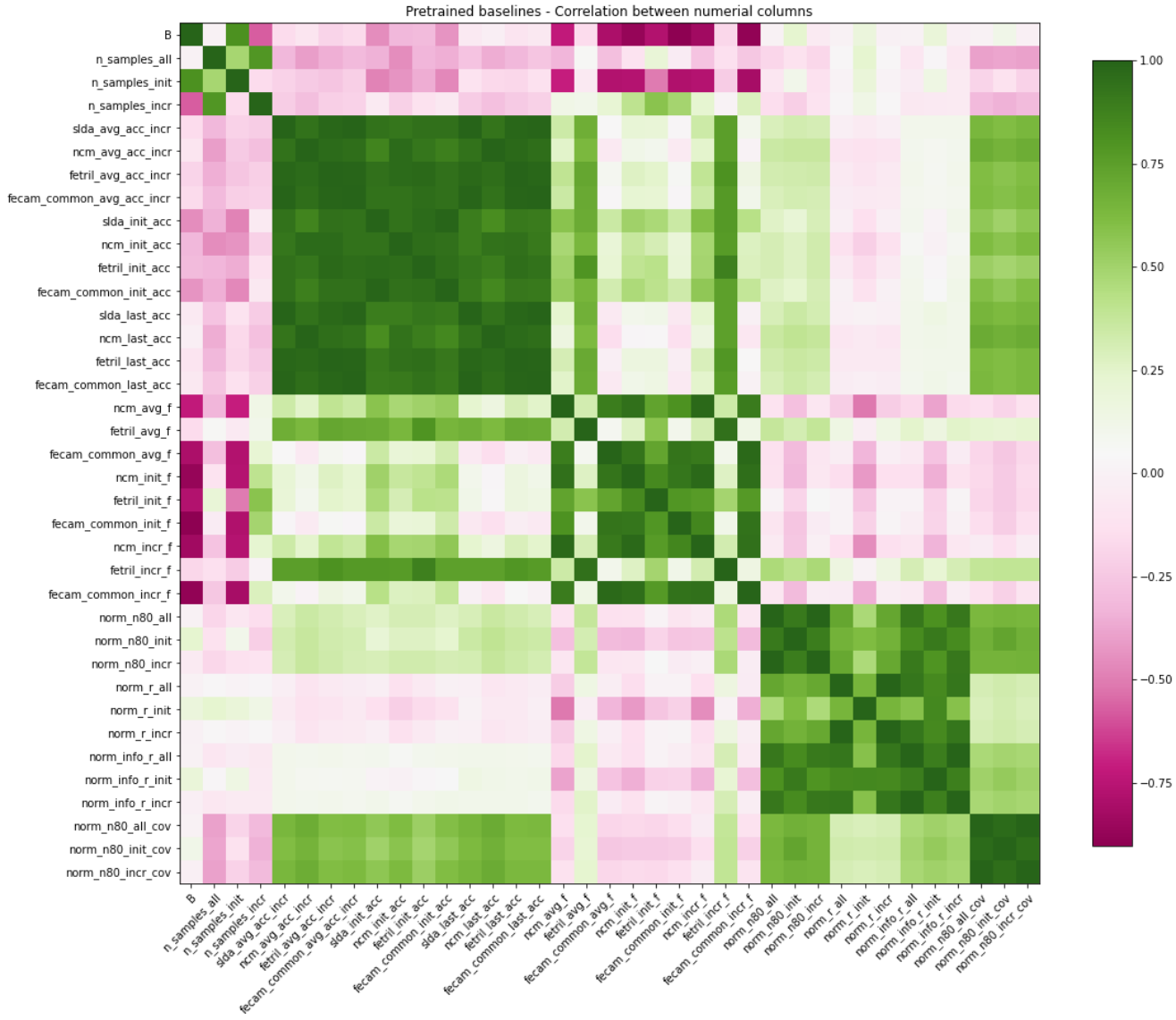

Figure 15. Pearson correlation coefficients for EFCIL experiments using a pre-trained baseline model as a feature extractor.

References

[1] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022.

[2] Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. RankMe: Assessing the downstream performance of pretrained self-supervised representations by their rank. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10929–10974. PMLR, 23–29 Jul 2023.

[3] Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost van de Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[4] Tyler L. Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 220–221, 2020.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2016.

[6] Christine Kaeser-Chen, Kiat Chuan Tan, Walter Reade, and Maggie Demkin. Herbarium 2020 - fgvc7, 2020.

[7] Minchul Kim, Feng Liu, Anil Jain, and Xiaoming Liu. Dcface: Synthetic face generation with dual condition diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12715–12725, 2023.

[8] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[9] Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning. *arXiv preprint arXiv:2202.00275*, 2022.

[10] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *ICCV*, pages 3476–3485. IEEE Computer Society, 2017.

[11] Aristeidis Panos, Yuriko Kobe, Daniel Olmeda Reino, Rahaf Aljundi, and Richard E. Turner. First session adaptation: A strong replay-free baseline for class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18820–18830, October 2023.

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[13] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490, 2012.

[14] Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3911–3920, January 2023.

[15] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[16] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Conference on Computer Vision and Pattern Recognition*, CVPR, 2017.

[17] Mert Bülent Sarıyıldız, Karteek Alahari, Diane Larlus, and Yannis Kalantidis. Fake it till you make it: Learning transferable representations from synthetic imagenet clones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8011–8021, 2023.

[18] Mert Bulent Sariyildiz, Yannis Kalantidis, Karteek Alahari, and Diane Larlus. No reason for no supervision: Improved generalization in supervised models. In *ICLR 2023-International Conference on Learning Representations*, pages 1–26, 2023.

[19] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[20] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018.

[21] Kan Wu, Houwen Peng, Zhenghong Zhou, Bin Xiao, Mengchen Liu, Lu Yuan, Hong Xuan, Michael Valenzuela, Xi (Stephen) Chen, Xinggang Wang, Hongyang Chao, and Han Hu. Tinyclip: Clip distillation via affinity mimicking and weight inheritance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 21970–21980, October 2023.

[22] D Yi. Learning face representation from scratch. *CoRR*, 1411:7923, 2014.