Our appendix is organized as follows:

1. Appendix A contains detailed experimental results in the "no-side information" case. We show that adding side-information outperforms no side-information in all cases, including when the embedding for "no-side information" has dimension equal to that of (side-information + embedding).
2. Appendix B shows performance of transformation with side-information on out-of-distribution datasets.
3. Appendix C shows a more detailed analysis of performance degradation of the Backwards Compatible Training (BCT) algorithm when applied to ϕ_{new} on ImageNet.
4. Appendix D reports hyperparameters and training details across all experiments.
5. Appendix E reports the results of our ablation for the transformation function on both capacity and loss function.
6. Appendix F reports details about embedding dimension ablations for ϕ_{new} and ϕ_{old} across datasets.
7. Appendix G shows our results with highly compressed side-information.
8. Appendix H shows our results on using FCT with a long sequence of small updates.
9. Appendix I shows our results on using FCT between models with disjoint training objectives.
10. Appendix J We report comparisons to [30] and [47], modified for our setting.
11. Appendix L reports the licenses of all the resources used.
12. Appendix K summarizes how, with a decentralized system design, the extra computational cost associated with FCT relative to BCT can be mitigated.

A. Comparison with No Side-Information

A.1. Standard Setup

Table 7 contains detailed results for "no side-information" in the standard setup. This corresponds to Tables 1, 2, and 3. In terms of implementation, we use $\psi(x) = 0$ as for our side-information function. We see that side-information improves model performance across all datasets. In particular, we see a CMC top-1 improvement of +3.2% on ImageNet, +1.0% on Places, and +1.0% on VGGFace2. The magnitude of improvement on $h(\phi_{old}/\psi)/h(\phi_{old}, \psi)$, a measure of how much our transformation improves the old features, is even greater, with CMC top-1 improvements of +7.4% on ImageNet, +2.1% on Places, and +3.2% on VGGFace2.

Finally, an argument could be made that the overall feature dimension is simply larger, when we consider side-information feature size together with the embedding dimension feature size. To show the benefit of our side-information over simply increasing the embedding dimension size, we show that a 256-dimensional feature vector for ImageNet only results in a 0.2% improvement in CMC top-1, far from the improvement which comes from having good side-information (+3.2%).

A.2. Sequence of Model Updates

Table 8 contains results corresponding to the sequence of model updates case (see Section 4.4) with no side-information. As in the previous section, this is implemented by setting $\psi(x) = 0$ for all inputs x. We denote this case $h(\phi)$ instead of $h(\phi, \psi)$ for embedding model ϕ and transformation model h. All other notation is consistent with Section 4.4.

B. Out-of-distribution Side-information performance

Side-Info	$(\phi_{new}/h(\phi_{old}, \psi))$ CMC top-1 — top-5 %	mAP@1.0
None (ϕ_{old})	14.5 — 35.2	3.7
None	16.9 — 40.7	5.6
Autoencoder	17.4 — 41.6	5.8
ϕ_{old}^{alt}	18.4 — 43.1	6.3
SimCLR	20.3 — 45.5	7.1
None (ϕ_{new})	21.9 — 46.8	7.1

Table 9. Using the same ImageNet embedding models ϕ_{old} and ϕ_{new} we analyze the same side-information strategies as Table 6 but instead evaluating on out of distribution data: the Places-365 validation set. We show that the same trends hold even out of distribution. Note that the retrieval performance is quite poor since ImageNet and Places-365 are very different domains.

Here we provide a more detailed ablation of the out-of-distribution retrieval performance for various side-information transformation methods originally presented in Section 4.3. We present these results in Table 9. We see the same trends from Table 6 repeated on this very different domain.

$\mathcal{D}_{old}$	$\mathcal{D}_{new}$	Embedding Dimension	$\phi_{new}/h(\phi_{old})$		$h(\phi_{old})/h(\phi_{old})$	
			CMC top 1—5 %	mAP %	CMC top 1—5 %	mAP %
ImageNet-500	ImageNet-1k	128	61.8 — 80.5	39.9	51.9 — 69.8	36.1
ImageNet-500	ImageNet-1k	256	62.0 — 80.9	38.7	53.8 — 71.9	35.8
Places-182	Places-365	512	34.8 — 63.6	17.5	31.9 — 60.4	16.4
VGGFace2-863	VGGFace2-8631	128	91.5 — 97.3	60.1	84.0 — 93.4	47.7

Table 7. Detailed results in the "no side-information" case. We see a substantial improvement in retrieval performance across all datasets.

Case	CMC top-1 — top-5 %	mAP@1.0
ϕ_1/ϕ_1	29.6 — 44.1	15.5
ϕ_2/ϕ_2	46.5 — 65.1	28.7
ϕ_3/ϕ_3	68.1 — 84.4	45.0
$\phi_2/h_2(\phi_1)$	36.5 — 53.5	23.7
$\phi_3/h_3(\phi_2)$	61.8 — 80.5	39.9
$\phi_3/h_3(h_2(\phi_1), g_2(\phi_1))$	44.9 — 68.0	26.4
$\phi_3/h_{1\to3}(\phi_1)$	53.9 — 74.2	30.6

Table 8. Model compatibility in sequence of updates with no side-information (see Table 5 for results with side-information). We see a large performance degradation $v_1 \to v_3$ versus with side-information (-6.0% CMC top-1) and an even larger degradation in the sequential case $v_1 \to v_2 \to v_3$ (-12.5% CMC top-1), showing that side-information is crucial for sequential updates to prevent feature drift.

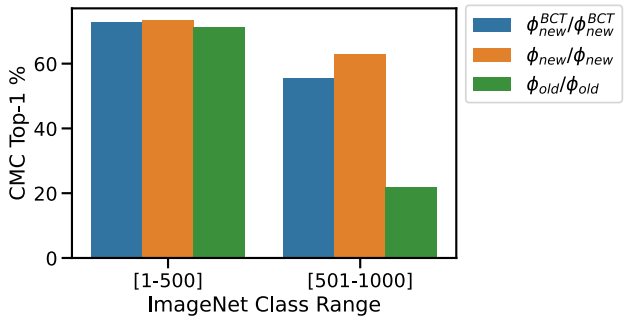

Figure 6. CMC Top-1 performance on the validation set of ImageNet for various cases by class range. Note that ϕ_{old} has not seen any data from classes [501-1000]. Note how ϕ_{new}^{BCT} performance is degraded primarily for the classes on which ϕ_{old} also performs poorly. This shows that the old model biases training of the new model for BCT.

C. BCT New Model Biased Performance Degradation

In Figure 6 we show how BCT training of the new model biases it towards the old model. Since the old model has not seen any data from ImageNet classes [501-1000], it naturally performs quite poorly on these classes. However, performance on classes [1-500] is quite similar across ϕ_{new} and ϕ_{old}. ϕ_{new}^{BCT} sees some improvement on classes [501-1000], however it performs 6.4% worse than ϕ_{new} on these classes. Given that its accuracy is similar to that of ϕ_{old} and ϕ_{new} on classes [1-500], we can conclude that the effect of BCT training is to bias ϕ_{new}^{BCT} to perform more similarly to ϕ_{old}, i.e. poorly on [501-1000]. Transferring unwanted biases from old to new model is a crucial drawback of the BCT method.

D. Architectures, Hyperparameters, and Training

D.1. Architecture Details

Embedding Models For our embedding model architectures, we use the standard ResNet [17] and MobileNetV1 [19] with penultimate layers modified to output 128 dimension embeddings (for VGGFace2 and ImageNet) and 512 dimension embeddings (for Places-365). We find that this architecture modification performs either the same or better than their higher dimension counterparts. We also observed that directly modifying the output dimension of the penultimate layer performs equivalently to projecting the output of the penultimate layer to a lower dimension, as in [44].

Transformation Models See Figure 4 for the basic transformation structure. The projection units in this diagram are MLPs with architecture: Linear($d \to 256$) $\to$ BatchNorm $\to$ ReLU $\to$ Linear($256 \to 256$) $\to$ BatchNorm $\to$ ReLU, where d is the dimension for either the embedding (d_{old}) or side-information (d_{side}). The linear layers use bias, however preliminary experiments have shown this is not necessary. The outputs of both projection branches are concatenated together before being passed to the "Mixer", which has architecture Linear($512 \to 2048$) $\to$ BatchNorm $\to$ ReLU $\to$ Linear($2048 \to 2048$) $\to$ BatchNorm $\to$ ReLU $\to$ Linear($2048 \to d_{new}$).

D.2. Model Selection Details

We did not perform hyperparameter tuning on transformation architecture training. We reused these ImageNet hyperparameters on the other datasets and embedding dimension sizes. The hyperparameters for ResNet and MobileNet training were taken from ResNetV1.5 [33], an improved training setup for ResNet. Hyperparameter details for this setup are provided in the in the individual dataset sections.

D.3. Hardware Details

We trained all of our models on 8 Nvidia V100 GPUs with batch size 1024. We have found that it is possible to decrease the batch size proportionally with the learning rate to use with fewer resources (e.g. batch size 1024 with learning rate 1.024 corresponds to batch size 256 with learning rate 0.256) without a drop in performance.

D.4. Side-Information

D.4.1 SimCLR

Architecture We use a standard ResNet50 [17] architecture with feature output directly modified to 64 or 128 for all of our SimCLR results. We add an extra BatchNorm layer to the end of the model, as suggested in [10].

Training Most of our training procedure is taken directly from the original SimCLR paper [9]. We use the multi-crop augmentation procedure originating from [6], specifically implemented as in [7].

D.4.2 Autoencoder

Training We train our AutoEncoder with the Adam optimizer using learning rate 3×10^{-4} and weight decay 0.0 for 100 epochs with cosine learning rate decay and 5 epochs of linear warmup with batch size 512.

D.5. ImageNet-1k Training

Transformation We train the transformation for 80 epochs with the Adam [21] optimizer. We use learning rate 5×10^{-4}, weight decay $3.0517578125 \times 10^{-5}$, cosine annealing learning rate schedule with one cycle [27] with linear warmup [16] for 5 epochs, taken from ResNetV1.5 [33]. At epoch 40 we freeze the BatchNorm statistics. We find empirically that this makes training more stable for smaller embedding sizes. We suspect that there is some configuration of hyperparameters and learning rates where this is not necessary, however we were not able to find it. For normalization methods with no batch statistics (e.g. LayerNorm [2]), we find that this is not necessary, but we get slightly worse performance (61.2% vs 61.6% CMC top-1 in the no side-information case with LayerNorm instead of BatchNorm).

Old and New Embedding Models We train the old and new embedding models (standard ResNet50 with 128 dimension embedding) with ResNetV1.5 hyperparameters [33]. We train with batch size 1024, learning rate 1.024, weight decay $3.0517578125 \times 10^{-5}$, momentum 0.875, and cosine learning rate decay with 5 epochs of linear warmup for 100 total epochs.

D.6. Places-365 Training

Transformation We train the Transformation for the same duration and with the same hyperparameters as for ImageNet.

Old and New Embedding Models We train the old and new embedding models for the same duration and with the same hyperparameters as for ImageNet. We use embedding dimension 512 for our ResNet50 models. We explain this choice in Appendix F.

D.7. VGGFace2 Training

Transformation We train the Transformation for the same duration and with the same hyperparameters as for ImageNet. Since the embeddings are normalized in this instance, we normalize the outputs of the Transformation during both training and inference.

Old and New Embedding Models We train VGGFace2 with the ArcFace [11] loss function. Following [11], we use a margin of 0.5 and scale of 64, with embedding dimension 128 (we find this to perform equally to embedding dimension 512, which was used in [11]). We also resize to 3x224x224 as we find this does better than 3x112x112, which is a standard for face retrieval applications. We train the old and new embedding models for the same duration and with the same hyperparameters as for ImageNet.

D.8. BCT Modifications

Old and New Embedding Models We use the author provided code [43] with a few modifications. In particular, we add the ResNetV1.5 parameters (stated previously) to properly compare with our method and modify the output embedding dimension to 128 (for ImageNet and VGGFace2) and 512 (for Places-182 and Places-365). We find that simply projecting the output layer performs equally well. These modifications result in significant improvement over the original code provided. In particular, ϕ_{new}^{BCT} / ϕ_{new}^{BCT} performance goes from 60.3% CMC top-1 before our modifications to 62.4% CMC top-1 after our modifications.

E. Transformation Size Ablation

E.1. Transformation Capacity and Training

The transformation function h should have small memory foot-print and computational cost. In Table 11, for the same setup as ImageNet experiment as in Section 4, we show accuracy of the transformed features for a different transformation model architectures with growing width. In Table 12 we compare effect of loss function. For KL-divergence, we apply both ϕ_{new} and $h(\phi_{old}, \psi)$ to the linear classifier trained with the new model (which is frozen), get log-probabilities, and then apply the Softmax function. We also considered reversed KL-Divergence. Empirically,

Embedding Size	ImageNet-{500, 1k}		Places-{182, 365}		VGGFace2-{836, 8631}	
	ϕ_{old}/ϕ_{old}	ϕ_{new}/ϕ_{new}	ϕ_{old}/ϕ_{old}	ϕ_{new}/ϕ_{new}	ϕ_{old}/ϕ_{old}	ϕ_{new}/ϕ_{new}
128	46.5	**68.0**	29.5	21.9	84.0	**96.6**
256	48.0	67.8	29.7	23.2	83.8	95.7
512	49.1	67.4	29.6	**37.0**	83.6	96.4
1024	49.0	67.1	29.5	36.7	84.1	95.9
2048	48.5	67.9	29.8	37.0	83.9	95.1

Table 10. Embedding dimension size ablation on ImageNet, Places-365, and VGGFace2 for CMC top-1. The ϕ_{old} architectures are ResNet50 for Places and ImageNet and ResNet18 for VGGFace2. The ϕ_{new} architectures are ResNet50 for all datasets. We train ϕ_{old} on ImageNet-500, Places-182, and VGGFace2-863 and ϕ_{new} on ImageNet-1k, Places-365, and VGGFace2-8631, respectively. We chose to use the dimension corresponding to the best performing ϕ_{new} for each dataset (bolded). In the case of a tie, we chose the lower dimensional model.

# of params (M)	$(\phi_{new}/h(\phi_{old}, \psi))$ CMC top-1 — top-5 %
0.79	62.9 — 81.0
1.9	64.1 — 81.8
5.7	65.0 — 82.3
19.6	65.0 — 82.5

Table 11. Effect of transformation function capacity on accuracy. Accuracy seems to saturate at a relatively small number of parameters. Note that there is not a one-to-one comparison between number of parameters and FLOPS, as convolutional layers tend to have a higher FLOPS to parameter count ratio. See Figure 3 for a direct comparison of these attributes.

Loss	$(\phi_{new}/h(\phi_{old}, \psi))$ CMC top-1 — top-5 %
MSE	65.0 — 82.3
KL	60.7 — 78.3
KL-Reversed	55.7 — 76.1

Table 12. Effect of loss function on training of FCT transformation.

MSE with target feature outperforms other choices. This has also been observed recently in [4].

F. Embedding Dimension ablation

In this section, we report the effect of embedding dimension in performance of embedding models for ImageNet, Places-365, and VGGFace2. The old and new embedding models' top-1 retrieval performance is shown for different embedding dimensions in Table 10. In all cases we directly modify the feature layer output size, rather than projecting the original higher dimension output (e.g., 2048-dimensionsonal features of ResNet50) to a lower dimension, as in [44]. Empirically, for ImageNet-1k and VGGFace2-8631 embedding dimension of 128 obtains highest accuracy, while for Places-365, an embedding dimension 512 performs the best. Interestingly, old model performance is superior to new model performance for Places-365 at embedding dimensions 128 and 256, validating the notion that we need a higher embedding dimension for that particular dataset.

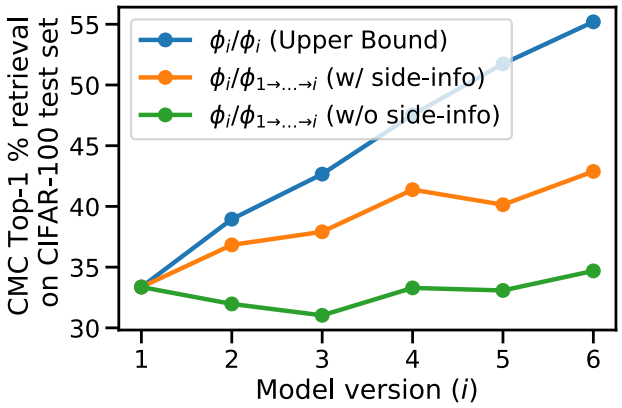

Figure 7. Sequence of updates between models trained on subsets of CIFAR-100 dataset with 50, 60, 70, 80, 90, and 100 classes.

G. Compressed SimCLR results

[38] presents a method which results in a highly compressed contrastive representation based on SimCLR with similar performance. We used the method in [38] to train a SimCLR model with feature dimension $\in \mathbb{Z}_2^{128}$ on ImageNet-500 to be used as side-information. This is a similar setup as in Table 1, but with a 32 times more compressed side-information feature vector. We report the numbers for this case in Table 13. We see slightly worse performance than with standard SimCLR (-0.9%), however this shows that side-information representations can be compressed while still maintaining favorable transformation properties.

Case	CMC top-1—5 (%)	mAP@1.0
ϕ_{old}/ϕ_{old}	21.9 — 46.8	7.09
ϕ_{new}/ϕ_{old}	0.3 — 1.5	0.12
ϕ_{new}/ϕ_{new}	37.0 — 65.1	17.0
$h(\phi_{old}, \psi)/h(\phi_{old}, \psi)$	31.7 — 59.5	16.1
$\phi_{new}/h(\phi_{old})$	33.3 — 62.3	16.9
$\phi_{new}/h(\phi_{old}, \psi)$	35.1 — 63.7	17.5

Table 14. FCT compatibility results when the old and new architectures are trained on completely disjoint objectives. ϕ_{old} is a ResNet50-128 trained on ImageNet-1k and ϕ_{new} is a ResNet50-512 trained on Places-365. Side-information is SimCLR trained on ImageNet-1k. FCT is able to get very close to the upper bound even in this challenging scenario.

Case	CMC top-1—5 (%)	mAP@1.0
$h(\phi_{old}, \psi)/h(\phi_{old}, \psi)$	57.2 — 74.7	40.0
$\phi_{new}/h(\phi_{old}, \psi)$	64.1 — 82.0	42.7

Table 13. Results for the ImageNet retrieval setup (see Section 4) with a highly compressed SimCLR representation [38].

H. Long Sequence of Small Updates

In Figure 7, we demonstrate the efficacy of our method for a series of small updates. Model ϕ_i is a ResNet50-128 trained on CIFAR-$(10i + 40)$, where CIFAR-N is a subset of CIFAR-100 including only the first N classes. Following our notation from Section 4.4, $\phi_i/\phi_{1\to\ldots\to i}$ means the query embedding uses ϕ_i for indexing and the gallery embedding is obtained with the chain of transformations $h_{i-1\to i} \circ h_{i-2\to i-1} \ldots \circ h_{1\to 2}(\phi_1, \psi_1)$, where each transformation $h_{i-1\to i}$ uses side-information $\psi_{1\to\ldots\to i-1}$. We show that we can achieve meaningful improvement in performance even after the model has drifted quite significantly from ϕ_1 through a chain of updates. Further, the importance of side-information for sequence is evidence, as without side-information, we sometimes fall short of backward compatibility as defined by [44].

I. ImageNet-1k to Places-365 evaluated on Places-365

In this scenario, the training objectives of ϕ_{old} and ϕ_{new} are completely disjoint. This means they share no training data in common. We train ϕ_{old} on ImageNet-1k and ϕ_{new} on Places-365. For side-information, we used SimCLR trained on ImageNet-1k. We use the Places-365 validation set as both the query and gallery sets for retrieval evaluation. We see that FCT is quite successful in this instance with full results presented in Table 14. We are able to maintain good model compatibility across different training objectives.

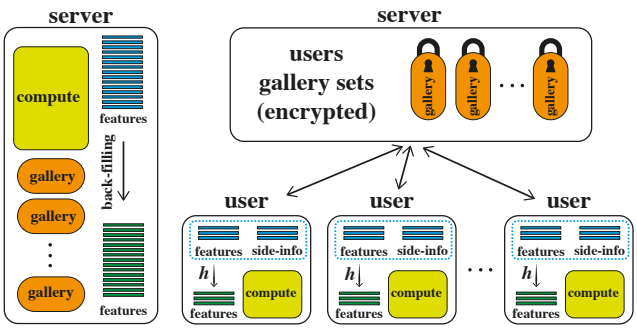

Figure 8. **(left)** A centralized system where data and compute reside on a server. Backfilling cost for such a system is mainly due to high compute. **(right)** A decentralized system where data is stored encrypted on a server, and FCT computations take place privately on edge devices. Storing all data on the user side is infeasible due to capacity constraints, so only side-info and embeddings are stored on device. The primary backfilling cost in this scenario is transferring images from the server to every edge device.

J. More Comparisons to Related Methods

In the absence of provided code, we have reimplemented [30] and [47] and modified their use for our setting. For [30], we modify the embedding dimensions to 128, change the loss function to cross entropy instead of ArcFace, and only train the forward transformation, but otherwise keep the original hyperparameters. For [47], we use their residual bottleneck transformation (RBT) architecture for forward transformation instead of our MLP. For old to new transform performance, RBT achieves 34.0% CMC top-1 ($\phi_{new}/h(\phi_{old})$ in our table) while LCE achieves 56.5%.

K. Decentralized Design

FCT-transformation computation is massively distributed on edge-devices.

Compared to BCT, FCT comes with two additional costs: (1) when a new image is being added to the gallery-set we need to compute and store side-information, and (2) when the embedding model is updated we need to perform a transformation. (1) is a one-time computation for every new image added to the system (an incremental cost) and the side-information is small relative to the size of the image. For (2) the transformation requires access to old embeddings and side-information. This is a small computation compared to full backfilling (i.e., running the new model on all images in the gallery set) as demonstrated in Figure 3.

For many applications FCT can be implemented in a decentralized fashion (Figure 8). In this setup, we have edge devices, each with their own gallery sets. When a new image is added, its embedding and side-information are computed and stored on device, and the raw image is encrypted and transferred to a remote server for storage. To

update the model from old to new, the FCT transformation runs on device. This design has three benefits: 1) The raw data always remains encrypted outside of the device without need to download it every time the model is updated. 2) Embedding and *side-information* computations are privately performed on-device one-time for every new image. 3) The small FCT transformation computation is massively distributed on edge-devices when updating the model.

L. Licenses

L.1. Software

PyTorch [36] is under the BSD license.

Python is under a Python foundation license (PSF)

L.2. Datasets

ImageNet [41] has no license attached.

Places-365 [52] is under the Creative Commons (CC BY) license.

VGGFace2 [5] has no license attached.