BYPASSING THE STABILITY-PLASTICITY TRADEOFF TO REDUCE PREDICTIVE CHURN

Anonymous authors
Paper under double-blind review

ABSTRACT

As more data is gathered over time, deployed ML models should be updated to take advantage of a larger sample size to improve performance. Unfortunately, even when model updates improve aggregate metrics such as accuracy, they can lead to errors on samples the previous model correctly predicted causing per sample regression in performance known as predictive churn. Such prediction flips erode user trust thereby reducing the effectiveness of the human-AI team as a whole. Current approaches for reducing predictive churn fall mainly into two categories: ensembles and distillation. While ensembles are the most effective, they require much greater resources for training, inference and storage. Distillation is much more efficient both in terms of training and inference, but is far less effective at reducing churn. We propose a missing middle-ground solution called StackEm based on accumulating models over time which achieves comparable performance to ensembles without any training time increases or changes to training procedures. Additionally, StackEm can be used to reduce churn for models which are already deployed, unlike ensembles. We demonstrate the effectiveness of StackEm on several computer vision benchmark datasets comparing against state-of-the-art churn reduction methods, showing comparable results to ensembles and a significant improvement over distillation.

1 INTRODUCTION

Model updates are necessary for many machine learning applications since learning on a larger number of samples as data is accumulated over time can often improve performance. Performance evaluation using common aggregate metrics such as accuracy can hide nuanced differences between the original model and updated model (Hossin & Sulaiman, 2015; Hardt et al., 2016). For example, models can make diverse errors at the sample level where switching from one model to another can cause perceived instability from the perspective of users even if overall the models perform the same, or the new model is better. It takes time for users to adapt to the strengths and weaknesses of a model, and a change in behaviour that violates their expectations is counterproductive as even a highly accurate model can have only marginal positive impact if it is not used to its full potential (Yin et al., 2019; Bansal et al., 2021). This is especially applicable in decision support settings where the user and model can be viewed as a team as opposed to APIs where a task is being fully automated. The flipping of predictions made by a new model relative to a base model is referred to as predictive churn (Fard et al., 2016). Not all churn is undesireable as prediction flips which result in correct classification are ideal, and flips between erroneous predictions are largely benign. Thus, we focus on **negative flips** or relevant churn: samples correctly predicted by the base model and incorrectly predicted by the new model.

Reducing negative flips has received more attention recently as the adoption of ML grows, with the two most common perspectives being variance reduction and prediction matching done by ensembles and distillation respectively. Ensembles aim to average out the stochastic aspects of neural network training resulting from random initialization, data augmentation, stochastic regularization techniques such as dropout, and the non-convex nature of the loss landscape (Zhao et al., 2022). They do so at a large computational cost since in order to observe reduced churn, the base model must be an ensemble, as does the new model, and this increases both training and inference costs linearly with the number of models in the ensemble. The assumption that the base model is an ensemble is impractical since any model that has already been deployed is not amenable to this technique. Distillation instead tries to balance learning on a combination of true labels and the base model's predictions thus introducing a stability-plasticity tradeoff. Focusing too much on matching base model predictions inhibits the ability to learn from new data, and disregarding the base model's predictions does not limit churn (Jiang et al., 2021).

Between the effectiveness of ensembles and the efficiency of distillation exists a compromise that does not increase training cost, does not require modifications to the training procedure, and makes no assumptions about the base model. We propose keeping both the base model and new model, and using stacking where a meta-model combines their outputs to create a final prediction for a given sample. In settings where model updates are infrequent, this method is more efficient in terms of both training and inference compared to ensembles, and is consistently more effective than distillation. We make the following contributions:

1. Explain the existence of negative flips through the perspective of inter-sample gradient compatibility. This suggests the need for a method that can explicitly remember the knowledge of the base model without constraining the learning of the new model (Section 3).
2. Introduce model stacking in the context of model updates with additional data gathered over time and propose several principled ways of combining base and new model outputs (Section 4).
3. Show superior performance against distillation, and comparable performance to ensembles at a fraction of the cost (Section 5).
4. Investigate what role model calibration plays in reducing churn (Section 5).

2 Related Works

Forgetting An important concept in the context of churn is catastrophic forgetting where updating on new data overwrites previously learned knowledge thereby causing negative flips. Catastrophic forgetting is typically discussed in the lifelong learning setting where a model is trained on one task A, and needs to learn a new task B under the assumption that data from task A is no longer available (Chen & Liu, 2018). Updating on data from the new task leads to forgetting on the previous task, and a variety of optimization schemes have been developed to prevent such forgetting. The tension that exists between keeping model parameters stable enough to perform well on task A while having enough flexibility to adapt to task B is known as the stability-plasticity tradeoff (Mermillod et al., 2013). Seminal work on preventing forgetting by Kirkpatrick et al. (2016) uses the Fisher Information matrix for task A to constrain the parameters learned on task B to be in a close neighbourhood to the parameters learned on Task A while taking into account individual parameter importance. Another line of work focuses on constraining gradients rather than distance in parameter space by projecting onto a subspace that improves both tasks, or at least does not decrease performance on task A (Farajtabar et al., 2019; Saha et al., 2021; Chaudhry et al., 2020). Other methods assume access to a buffer of past experiences, and replay experiences to reinforce original predictions (Verwimp et al., 2021). More recently Buzzega et al. (2020) combined experience replay with knowledge distillation to more effectively force the new model to make the same predictions on samples from the original task. Outside of lifelong learning, Toneva et al. (2018) introduced the concept of forgetting due to the stochastic nature of neural network optimization, where predictions can flip from right to wrong several times during the course of training. We discuss the implications of this work for churn and use it to motivate StackEm in Section 3.

Negative Flip Reduction Predictive churn has been mentioned in the literature under other terms such as performance regression and model backward compatibility. The concept was formally introduced by Fard et al. (2016) who focused on the model adaptation notion of churn which occurs when learning a new model on additional data. Two additional notions of churn exist: reproducing results across random initializations (variance), and changing the model architecture/hyperparameters (model upgrade). What separates the variance notion of churn from model adaptation and model upgrade is that it focuses only on stability, whereas the other two have a stability-plasticity tradeoff. Thus, the most effective method for reducing churn due to variance in the optimization procedure is not the most effective for the other two types of churn. We focus on the model adaptation notion of churn in our work as we are interested in learning a new model with the same architecture as the base model while leveraging additional data.

The most efficient class of methods for reducing churn are based on distillation (Hinton et al., 2015) where the opitmization objective is modified to include a term that biases the predictions of the new model towards those of the base model. Fard et al. (2016) introduced the general concept of a stabilization operator meant to allow learning on new data while keeping predictions consistent with the base model. The actual stabilization operator used is referred to as anchor loss by other works, and is a distillation-based objective which trains on a mixture distribution of the ground truth one-hot targets and predictions made by the base model, similar to label smoothing (Müller et al., 2019). Anil et al. (2018) introduced co-distillation for reducing the variance notion of churn such that predictions are reproducible even when it is not feasible to control the random initialization. Co-distillation works by training 2 models in parallel using each other's predictions for distillation, and only one of the models is kept upon convergence. This procedure is then repeated, again keeping just one of the models upon convergence such that the churn between the two runs is significantly less than when training two models independently from different initializations. Bhojanapalli et al. (2021) built upon co-distillation by combining it with entropy regularization to achieve further churn reduction. Both co-distillation and anchor loss modify the training procedure for the base model which is impractical for models that are already in deployment. Jiang et al. (2021) address this limitation with an objective similar to anchor loss that requires fewer hyperparameters and achieves SOTA churn reduction in the model adaptation setting.

Yan et al. (2020) introduced a new distillation objective called focal loss where samples correctly predicted by the base model undergo a stronger distillation loss than other samples. Even though focal loss is meant to increase the plasticity of the new model, Yan et al. (2020) observed that ensembles have superior performance. Bahri & Jiang (2021) also showed that ensembles, though impractical due to their high training and inference costs, are by far the most effective way of reducing churn. In an attempt to reduce ensemble inference cost, Zhao et al. (2022) distill the knowledge from the average logits of an ensemble to a single model. Their method ELODI was shown to be more efficient, but not quite as effective in both accuracy and churn reduction as ensembles. It also requires training an ensemble for both the base and new model prior to applying distillation, so it is still very expensive relative to distillation, and is not applicable to already deployed models. Most similar to our approach is that of Cai et al. (2022) who show that storing models over time is effective for reducing churn in the structured prediction setting for both syntactic and semantic parsing tasks. In particular, they focus on the case of switching from one type of parser to another (model upgrade), rather than our setting of accumulating more data over time (model adaptation). A re-ranking procedure is used where the new model generates a set of candidate predictions, and the base model chooses from these predictions. However, re-ranking is a fundamentally different procedure compared to model stacking and is not directly transferable to classification tasks since the notion of candidate prediction does not exist in classification.

3 Understanding Churn

We are interested in the supervised classification setting where $\mathcal{D}_{\text{train}} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n_{\text{train}}}$ is our initial training data comprised of samples $x \in \mathbb{R}^d$, $y \in [k]$. We would like to learn the parameters θ of a model $f(\cdot;\theta)$ where $f : \mathbb{R}^d \to \mathbb{R}^k$. Let ℓ be the cross-entropy loss, and $\phi : \mathbb{R}^k \to \Delta_k$ be the Softmax function mapping from logits to the $k-1$ dimensional simplex. The parameters of the base model θ_{base} are obtained via empirical risk minimization

$$\theta_{\text{base}} = \arg\min_{\theta} \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} \ell(\phi(f(x;\theta)), y)$$

where ℓ is the cross entropy loss function.

Given additional data $\mathcal{D}_{\text{update}}$ we would like to learn new parameters θ_{new} such that the churn between θ_{base} and θ_{new} is less than a tolerance ϵ. Specifically, let $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{update}}$, and $\sigma : \mathbb{R}^k \to [k]$ be the $\arg\max$ function mapping from soft model scores to a hard prediction. The churn-constrained optimization problem is formulated as follows (Jiang et al., 2021)

$$\min_{\theta_{\text{new}}} \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell\left(\phi\left(f(x;\theta_{\text{new}})\right), y\right) \ \text{ s.t. } \ C(\theta_{\text{base}}, \theta_{\text{new}}) < \epsilon$$

$$C(\theta_{\text{base}}, \theta_{\text{new}}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \mathbb{1}[\sigma(f(x;\theta_{\text{base}})) \neq \sigma(f(x;\theta_{\text{new}}))]$$

Note that we use a hard definition of churn rather than the soft, divergence-based churn used by (Jiang et al., 2021).

$C(\theta_{\text{base}}, \theta_{\text{new}})$ counts samples that both θ_{base} and θ_{new} predict incorrectly, what we refer to as benign flips. We believe negative flips to be most disruptive to user-model workflows, so we consider the notion of *relevant churn* that focuses on negative flips instead

$$C_{\text{rel}}(\theta_{\text{base}}, \theta_{\text{new}}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \mathbb{1}[\sigma(f(x;\theta_{\text{new}})) \neq \sigma(f(x;\theta_{\text{base}})) \wedge \sigma(f(x,\theta_{\text{base}})) = y] \tag{1}$$

as in (Yan et al., 2020).

Jiang et al. showed that the above constraint optimization problem is equivalent to distillation by training with a mixture distribution of the ground truth one-hot target and output probability from the base model

$$\min_{\theta_{\text{new}}} \frac{1-\alpha}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(\phi(f(x;\theta_{\text{new}})), y) + \frac{\alpha}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(\phi(f(x;\theta_{\text{new}})), \phi(f(x;\theta_{\text{base}})))$$

Distillation introduces an explicit stability-plasticity tradeoff between learning on new data and matching the predictions of the base model which can be controlled by varying α. It allows for substantial churn reduction and is SOTA among non-ensemble based methods Jiang et al. (2021), but to achieve further churn reduction requires limiting the performance of $f(\cdot;\theta_{\text{new}})$.

While churn is defined between two models, it is important to note that it also occurs during the course of training a single model. Toneva et al. (2018) introduced the notion of a forgetting event where if $\hat{y}_i^t = \arg\max_k f(x_i, \theta^t)$ and $\text{acc}_i^t = \mathbb{1}[\hat{y}_i^t = y_i]$, then a forgetting event occurs at time $t+1$ if $\text{acc}_i^t > \text{acc}_i^{t+1}$. On CIFAR-10, it was discovered that only $\sim$ 15k samples are unforgettable (i.e. no forgetting event occurs once correctly predicted the first time), meaning that the remaining $\sim$ 35k samples have unstable predictions during training. This instability between points along an optimization trajectory suggests churn can be attributed to some samples being incompatible with a given update. We define sample compatibility as the cosine similarity between the average batch gradient and individual sample gradient

$$g_{\mathcal{B}} = \nabla_\theta \frac{1}{|\mathcal{B}|} \sum_{(x,y)\in\mathcal{B}} \ell(\phi(f(x;\theta)), y) \qquad g = \nabla_\theta \ell(\phi(f(x;\theta)), y)$$

$$\text{comp}(g_{\mathcal{B}}, g) := \frac{\langle g_{\text{B}}, g\rangle}{||g_{\mathcal{B}}||\,||g||}$$

The set of incompatible samples is then $\mathcal{S}_{\text{inc}} = \{(x,y) \in \mathcal{D} | \text{comp}(g_{\mathcal{B}}, g) < 0\}$, i.e. samples that incur an increase in loss when taking a step in the direction of the average gradient. Figure 1 shows the distribution of cosine similarities between the average gradient and per-sample gradients at 3 checkpoints during the training of a ResNet18 on CIFAR10. Even as the model fits the training data increasingly well, there are still many incompatible samples. Incompatible samples close to the decision boundary are precisely those that are likely to have a negative flip during training. As a proxy for distance to decision boundary, we use the margin as defined by (Pleiss et al., 2020) $M(x,y) = f_y(x) - \max_{i\neq y} f_i(x)$. The margin is affected by both the given sample's gradient, as well as the gradients of all other samples in the batch, where the former decreases the margin given a small enough learning rate, and the latter depends on how the knowledge from remaining samples generalizes to the current sample. We also compute gradient norms to see if the samples in $\mathcal{S}_{\text{inc}}$ have a significant impact on the overall batch weight update. Figure 2 compares gradient norm and

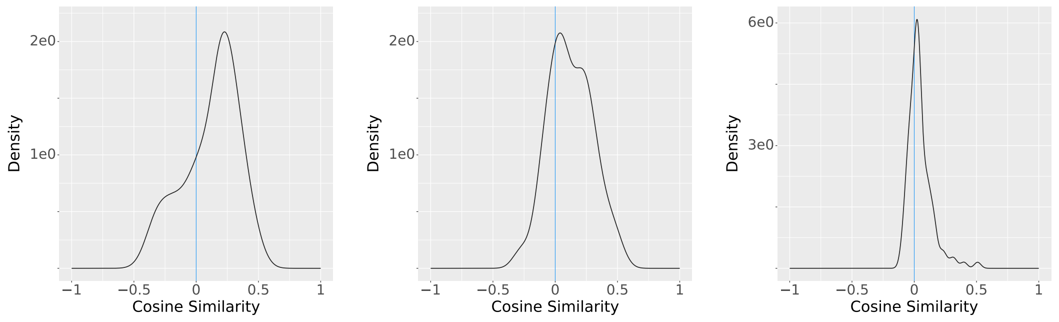


Figure 1: Gradient cosine similarity distribution throughout training on a random subset of 100 samples for a ResNet18 model trained on CIFAR10 data. Density to the left of the blue line corresponds to incompatible samples that will have increased loss after a weight update. Left, middle, and right figure correspond to epochs 10, 100, and 200 respectively.

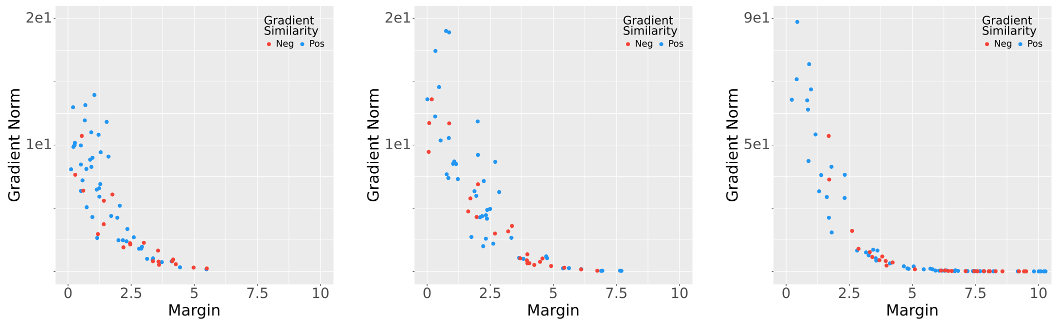


Figure 2: Gradient norm vs. sample margin grouped by gradient similarity revealing that even as the model fits the data well, there are still incompatible samples with both high gradient norm and small margin. Red points with low margin are at high risk of having a flipped prediction. Left, middle, and right figure correspond to epochs 10, 100, and 200 respectively.

margin for compatible (blue) and incompatible (red) samples. Incompatible samples with both high gradient norm, and small margin persist throughout training suggesting that forgetting on training samples is common until the final loss is 0. The churn reduction effectiveness of a regularization method which trains only a single model is thus limited by its ability to ensure self-consistency from one epoch to the next. Specifically, this means the ability to find a direction in weight space which simultaneously decreases the loss on all samples at every parameter update.

Thus, the performance of any single-model churn reduction method, of which distillation is a prime example, is limited by the existence of incompatible samples. In order to bypass the stability-plasticity tradeoff, we propose introducing a memory mechanism through model stacking that allows keeping the new model's predictions when it is more likely to be correct than the base model.

4 Stacking Models

We use f_{base} and f_{new} to denote $f(;\theta_{\text{base}})$ and $f(;\theta_{\text{new}})$ for cleaner notation. The ensemble approach for reducing churn (Figure 3) requires that both the base and new model are ensembles: $\mathcal{F}_{\text{base}} = \{f^{(1)}_{\text{base}}, ..., f^{(M)}_{\text{base}}\}$, $\mathcal{F}_{\text{new}} = \{f^{(1)}_{\text{new}}, ..., f^{(M)}_{\text{new}}\}$, where M is the number of models per ensemble, and inference is done by averaging the logits of the ensemble predictions

$$\mathcal{F}(x) = \frac{1}{M}\sum_{i=1}^{M} f^{(i)}(x)$$

This has two major challenges which limit its feasibility in practice. First, it increases both training and inference costs by a factor of M which can be prohibitively expensive. Second, since the base model needs to be an ensemble, it is not possible to apply this approach to an already deployed model.

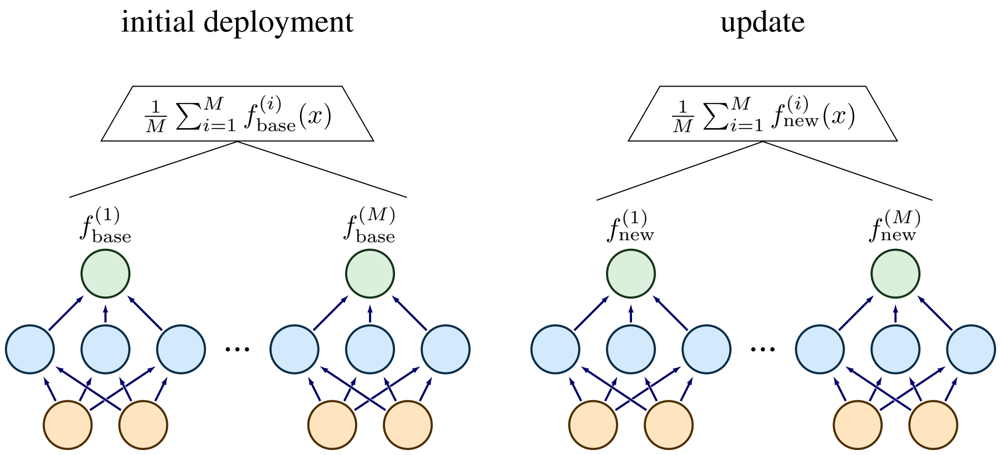


Figure 3: Traditional Ensemble approach for reducing churn. Both the base model and new model are an ensemble of M models in order to achieve sufficient variance reduction.

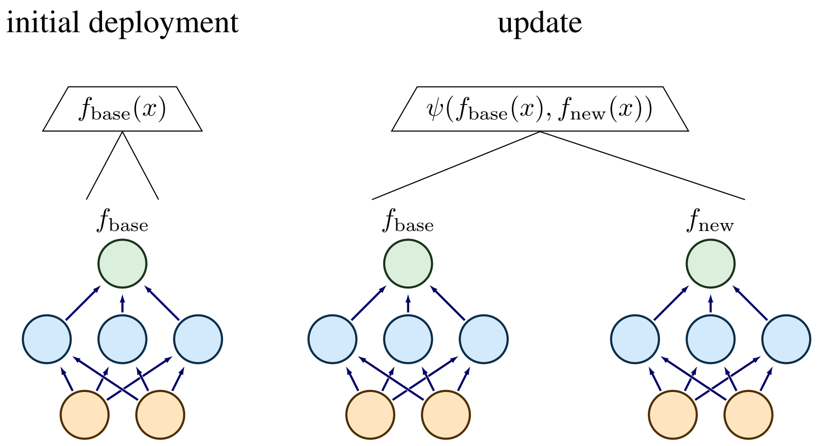


Figure 4: StackEm for reducing churn. After updating, a combination of the base and new model's outputs are used to generate the final prediction.

To bypass these limitations, we introduce StackEm (Figure 4) : a stacking based approach that fuses the output of f_{base} and f_{new} using a meta model ψ to generate the final prediction. While stacking has been around for decades (Wolpert, 1992), its use has not been explored for churn reduction, and the meta-models we consider are meant to reduce churn rather than increase accuracy (Ting & Witten, 1997). By not requiring f_{base} and f_{new} to be ensembles, our method would train $M-1$ models fewer than ensemble approach at each update. Most of all, it allows for churn reduction even for an already deployed base model. Ensembles and StackEm are not only different in terms of their computational requirements, but also in the way that they reduce churn.

Since ensembles results in both a more accurate base model and new model, most of the churn reduction advantages come from the improvement in performance, with some additional benefit coming from less variance in predictions as shown by Zhao et al. (2022). Under some strong assumptions, namely that the output logits are normally distributed, the authors show that

$$\mathcal{F}(X) = \frac{1}{M}\sum_{i=1}^{M} f^{(i)}(X) \sim \mathcal{N}(\boldsymbol{\mu}, \frac{1}{M}\boldsymbol{\Sigma})$$

assuming $f^{(i)}(X) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $X \sim P(X)$. We use capital X to emphasize that we are considering model outputs for a random input from the data distribution, not a particular realization. Thus, the difference between the logits of base and new ensembles is

$$\mathcal{F}_{\text{base}}(X) - \mathcal{F}_{\text{new}}(X) \sim \mathcal{N}(\boldsymbol{\mu}_{\text{base}} - \boldsymbol{\mu}_{\text{new}}, \frac{2}{M}(\boldsymbol{\Sigma}_{\text{base}} + \boldsymbol{\Sigma}_{\text{new}}))$$

which has a variance smaller by a factor of $\frac{1}{M}$ compared to single models.

Instead of reducing variance, StackEm aims to deviate from the base model for a sample x when the new model is more likely to be correct. By keeping both f_{base} and f_{new}, we can avoid creating a tradeoff between stability and plasticity. To show this, assume that f_{new} and f_{base} are perfectly calibrated. This means that $\mathbf{Pr}(\hat{Y} = Y | \hat{P} = p) = p, \forall p \in [0, 1]$ where $\hat{Y} = \arg\max_k f(X)_k$ (hard prediction) and $\hat{P} = \max_k \phi(f(X))_k$ (predicted probability). We capitalize the symbols to indicate that they are random variables jointly distributed according to $P(X, Y)$.

Perfect calibration implies that model accuracy $\text{acc}(f) = \mathbb{E}_{\hat{p}}[\mathbf{Pr}(\hat{Y} = Y | \hat{P} = p)]$ where the expectation is taken w.r.t. the distribution of predicted probabilities $\hat{P}$. If we have a meta-model ψ that chooses the highest confidence model according to the rule $\psi(x) = f_*(x)$ where $* = \arg\max_{\omega \in \{\text{base,new}\}} \max_c f_\omega(x)_c$, then ψ is also perfectly calibrated. We have

$$\begin{aligned}
\text{acc}(\psi) &= \mathbb{E}_{\hat{P}_\psi}[\mathbf{Pr}(\hat{Y}_\psi = Y | \hat{P}_\psi = p)] \\
&= \mathbb{E}_X\left[\max_k \phi\left(f_*(x)\right)_k\right] \qquad \# \text{ due to calibration} \\
&\geq \mathbb{E}_X\left[\max_k \phi\left(f_{\text{new}}(x)\right)_k\right] \\
&= \mathbb{E}_{\hat{P}_{\text{new}}}[\mathbf{Pr}(\hat{Y}_{\text{new}} = Y | \hat{P}_{\text{new}} = p)] \\
&= \text{acc}(f_{\text{new}})
\end{aligned}$$

and trivially $C_{\text{rel}}(\psi) \leq C_{\text{rel}}(f_{\text{new}})$ (see eq 1) since negative flips can only be reduced by using f_{base} instead of f_{new}. Thus, StackEm can reduce C_{rel} without decreasing accuracy, hence it bypasses the stability-plasticity tradeoff. We refer to the above ψ as **Confidence** in the results since it chooses the highest confidence prediction of the two models. We consider two other meta-models ψ to aggregate the outputs of f_{base} and f_{new} and compare their performance in the results section.

4.1 KL-DIVERGENCE

The KL-divergence between a model's output probabilities and the uniform distribution has been observed to be larger for in-distribution data compared to out-of-distribution data (Huang et al., 2021). While correlated with prediction confidence, this captures uniformity among remaining classes and prefers that the remaining probability is concentrated among a few classes. We thus choose the model that has the highest such KL-divergence for a given sample

$$\mathbf{u} = \left[\frac{1}{K}, ..., \frac{1}{K}\right] \in \mathbb{R}^K \qquad G_\omega(x) = D_{\text{kl}}(\mathbf{u} || f_\omega(x)) \qquad * = \underset{\omega \in \{\text{base,new}\}}{\arg\max}\, G_\omega(x)$$

$$\psi(x) = f_*(x)$$

4.2 LEARNED COMBINATION

Learning to use the outputs of f_{base} and f_{new} to generate a new prediction capable of correcting the mistakes of both models is the original approach used for stacking (Ting & Witten, 1997). A model $h^* : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^k$ is learned to maximize the accuracy on the validation set given the outputs of f_{base} and f_{new}:

$$h^* = \underset{h \in \mathcal{H}}{\arg\min} \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(x,y) \in \mathcal{D}_{\text{val}}} \ell(h(f_{\text{base}}(x), f_{\text{new}}(x)), y)$$

$$\psi(x) = h^*(f_{\text{base}}(x), f_{\text{new}}(x))$$

where $\mathcal{H}$ is the hypothesis space for the learned combination. We perform 5-fold cross validation to select the the best $h \in \mathcal{H}$ as to avoid overfitting.

5 Experiments

Data We largely follow the experimental setup of Jiang et al. (2021) and focus on benchmark computer image classification datasets. However, rather than using small subsets of data for training and updates (1000 samples for both) as in Jiang et al. (2021), we use all available data in order to obtain results that are representative of real-world models. For all datasets, we use a 60/20/20 split into training/validation/update partitions which enables a clear performance improvement for f_{new} relative to f_{base} since it is trained on significantly more samples. All of the datasets used have a pre-defined test partition which we use for final evaluation. We use the following datasets:

1. CIFAR10, CIFAR100 (Krizhevsky), STL10 (Coates et al., 2011) using ResNet18 (He et al., 2015)
2. MNIST (LeCun & Cortes, 2010), EMINST (Cohen et al., 2017), KMNIST (Clanuwat et al., 2018), FashionMNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011) using LeNet-5 (Lecun et al., 1998)

Training: We use early stopping like Jiang et al. (2021) to avoid having to find an optimal fixed number of epochs for each dataset/method combination which is complicated by the fact that f_{new} has access to more data and would require a different number of update iterations than f_{base}. f_{base} is trained on $\mathcal{D}_{\text{train}}$, and f_{new} is trained from scratch on $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{update}}$. We use the Adam optimizer with a learning rate of 0.001 for both models. Data augmentation in the form of random horizontal flips and crops is used for the CIFAR10, CIFAR100, and STL10 datasets (Shorten & Khoshgoftaar, 2019). Every set of experiments is run with 10 random seeds to obtain a reliable estimate of average accuracy and C_{rel} (eq1). We note that since we are randomly splitting the original training sets into train/validation/update, the aim is not to reach SOTA accuracy, rather an effective churn reduction method should be as accurate as the baseline model while significantly reducing C_{rel}.

Jiang et al. (2021) investigated several churn reduction baselines and found that distillation outperforms all of them including initializing f_{new} with θ_{base}, label smoothing Bahri & Jiang (2021), and Mix-Up (Zhang et al., 2017). We use the Distillation approach from (Jiang et al., 2021) shown in Section 3 as the SOTA distillation approach to compare against, and use the same search space for the hyperparameter $\alpha \in \{0.1, 0.2, ..., 0.9\}$ as in their paper for a fair comparison. For Ensemble, we consider $M \in \{3, 5, 7\}$ as a larger number of models results in diminishing variance reduction relative to the increase in training costs. Finally, for StackEm only the Learned Combination requires hyperparameter tuning, namely a search over the space of models used to combine the outputs of f_{base} and f_{new}. We consider logistic regression, random forests, and gradient boosting models, with the hyperparameters used for each listed in the appendix.

Table 1 compares the performance of the considered methods, and results for two additional baselines can be seen in the appendix. The best performing method for each dataset is bolded, and since we are looking at both accuracy and C_{rel} (relevant churn), each row has 2 bold entries. If the observed difference in C_{rel} is not statistically significant (one-sided t-test with $\alpha = 0.05$), we highlight both methods. In all cases, StackEm outperforms Distillation by a significant margin, and the reduction in churn is greater than the increase in accuracy. This is not the case with Ensemble where relative to Distillation, churn reduction is less than the observed increase in accuracy. On Fashion-MNIST and CIFAR100 StackEm even outperforms Ensemble. While Ensemble achieves less churn than StackEm for the remaining datasets, the results are not statistically significant as confirmed by a t-test. We emphasize that StackEm achieves nearly as much churn reduction as Ensemble at a fraction of the cost, in this case 7x less memory, training compute cost, and inference cost. Due to limited space, confidence intervals are not included in Table 1, but boxplots can be found in the Appendix.

Given the substantial churn reduction performance of StackEm , we investigate which meta-model is best. Table 2 shows that on all datasets except MNIST and EMNIST, Confidence is consistently the most effective at reducing churn, even though it does not achieve the highest accuracy. To understand this discrepancy between accuracy and churn, Figure 5 shows the number of negative and positive

Table 1: Comparison of churn reduction methods on benchmark image classification datasets. StackEm outperforms Distillation on both accuracy and C_{rel} (relevant churn) across all datatsets, and even outperforms Ensemble in terms of reducing C_{rel} on FashionMNIST and CIFAR100. Note that we report C_{rel} as a percentage, hence why it takes on values greater than 1.

	No Regularization		Distillation		Ensemble (7x)		StackEm	
Dataset	Acc $\uparrow$	$C_{\text{rel}} \downarrow$	Acc $\uparrow$	$C_{\text{rel}} \downarrow$	Acc $\uparrow$	$C_{\text{rel}} \downarrow$	Acc $\uparrow$	$C_{\text{rel}} \downarrow$
MNIST	98.958	0.640	99.072	0.410	**99.520**	**0.128**	99.374	0.162
EMNIST	93.490	2.359	93.647	1.656	**94.980**	**0.744**	94.279	**0.760**
KMNIST	95.405	1.991	95.585	1.595	**97.119**	**0.534**	96.124	**0.607**
FashionMNIST	90.601	3.642	90.653	2.634	**92.699**	**1.353**	91.509	**1.312**
SVHN	87.319	6.096	89.002	3.927	**92.981**	**1.557**	90.013	**1.685**
CIFAR10	82.598	6.450	83.372	4.354	**85.226**	**2.399**	84.130	**2.526**
CIFAR100	49.851	9.745	50.613	5.814	**54.371**	4.568	51.524	**4.092**
STL10	64.140	11.672	64.762	6.592	**71.678**	**4.614**	67.174	**4.971**

Table 2: Comparison of the three meta-models ψ considered for StackEm . Learned Model is the most accurate except for on CIFAR100 where it fails likely due to the low accuracy of both f_{base} and f_{new}. However, Learned Model does worse than Confidence for churn reduction except on MNIST.

	Confidence		KL-Div		Learned Model	
Dataset	Acc $\uparrow$	$C_{\text{rel}} \downarrow$	Acc $\uparrow$	$C_{\text{rel}} \downarrow$	Acc $\uparrow$	$C_{\text{rel}} \downarrow$
MNIST	99.271	0.183	99.194	0.242	**99.374**	**0.162**
EMNIST	94.279	**0.760**	93.947	1.120	**94.391**	1.095
KMNIST	96.124	**0.607**	95.430	0.760	**96.359**	0.679
FashionMNIST	91.509	**1.312**	90.949	1.901	**91.760**	1.550
SVHN	90.013	**1.685**	89.724	1.867	**90.809**	1.820
CIFAR10	84.130	**2.526**	83.316	3.371	**84.839**	2.618
CIFAR100	**51.524**	**4.092**	50.066	5.473	46.047	11.620
STL10	67.174	**4.971**	66.000	7.200	**68.504**	6.447

flips made by each meta-model. For Confidence and Entropy, ψ is not capable of making additional negative or positive flips compared to just using f_{new} since ψ chooses between f_{base} and f_{new}. However, Learned Model is capable of making new predictions that differ from both f_{base} and f_{new}, so while it is more effective at reducing negative flips relative to Confidence (green bar for Learned Model is lower than Confidence), it introduces new errors on samples that both f_{base} and f_{new} correctly predict which results in a higher total number of negative flips compared to Confidence.

The Confidence meta-model works best under the assumption that predictions are calibrated, so we investigate if further churn reduction is achievable by improving calibration. The calibration of both models before and after temperature scaling can be seen in Figure 6 for a ResNet18 models on CIFAR10. Both expected calibration error (ECE) and maximum calibration error (MCE) are significantly reduced through temperature scaling (Guo et al., 2017). Surprisingly, this does not help reduce churn, or improve model accuracy. Accuracy without scaling is 84.17% and after scaling it is 84.08% while C_{rel} with scaling is 2.17% without scaling and 2.19% with scaling. We observe that both f_{base} and f_{new} were systematically overconfident in their predictions prior to temperature scaling.

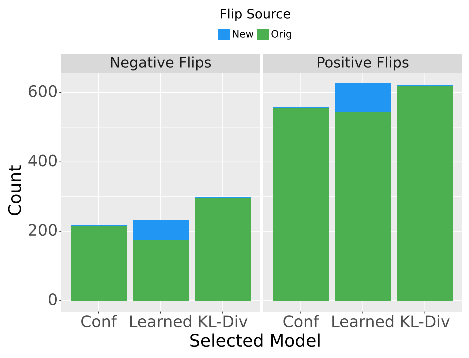
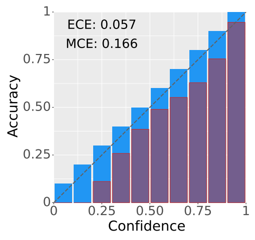
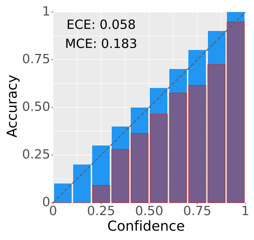


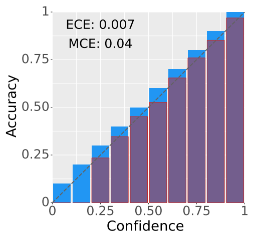
Figure 5: Flip counts for the 3 meta-models for ResNet18 models trained on CIFAR10. Confidence has the fewest negative flips. Learned and KL-Div have more positive flips, showing that better accuracy does not imply less churn.



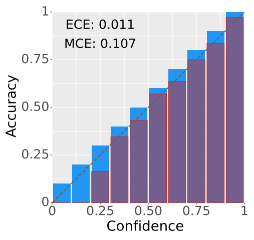
(a) f_{base} Uncalibrated



(b) f_{new} Uncalibrated



(c) f_{base} Calibrated



(d) f_{new} Calibrated

Figure 6: Calibration of ResNet18 models trained on CIFAR10.

Table 3: Changes in prediction confidence ranking due to temperature scaling.

Switch To	Benign	Good	Bad
f_{base}	234	9	18
f_{new}	41	11	11

Thus, improved calibration does not affect the ranking in prediction confidence between the models for many samples. Table 3 shows that calibration results in a total of just 324 total changes in ranking between f_{base} and f_{new}. Most of the changes are benign meaning they occur for samples which f_{base} and f_{new} both get right or both get wrong. Good switches to f_{base} correspond to decreases in negative flips, and bad switches to f_{new} are increases in negative flips. The Appendix shows the distribution of prediction confidence on negative flips focusing on samples that f_{new} predicts with higher confidence than f_{base} and thus cannot be eliminated using the Confidence meta-model. Perfect churn reduction can thus occur only when using a meta-model that sometimes chooses f_{base} even when $\max_k f_{\text{base}}(x)_k < \max_k f_{\text{new}}(x)_k$.

6 Discussion and Limitations

By showing that prediction instability exists due to incompatibility between samples at different steps during the learning process, we motivate the need for a churn reduction method that does not rely on the new model having to match the predictions of the base model. We showed that our method StackEm is capable of bypassing the stability-plasticity tradeoff by reverting to the base model when necessary for stability, and letting the new model be unconstrained for maximum plasticity. While the performance advantages over distillation are clear, the cost of inference becomes $O(T)$ where T is the number of model updates performed. Thus, StackEm eventually suffers the same computational drawbacks as ensembles at inference time when model updates are frequent, though training costs are still greatly reduced. Overall, StackEm gives ML practitioners an option that is in between the two extremes of ensembles and distillation. It can serve as an easy to implement way of maintaining user trust throughout model deployment.

References

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. April 2018.

Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Adv. Neural Inf. Process. Syst.*, 33:3884–3894, 2020.

Dara Bahri and Heinrich Jiang. Locally adaptive label smoothing for predictive churn. February 2021.

Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. Does the whole exceed its parts? the effect of ai explanations on complementary team performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–16, 2021.

Srinadh Bhojanapalli, Kimberly Wilber, Andreas Veit, Ankit Singh Rawat, Seungyeon Kim, Aditya Menon, and Sanjiv Kumar. On the reproducibility of neural network predictions. February 2021.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. April 2020.

Deng Cai, Elman Mansimov, Yi-An Lai, Yixuan Su, Lei Shu, and Yi Zhang. Measuring and reducing model update regression in structured prediction for NLP. February 2022.

Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr. Continual learning in low-rank orthogonal subspaces. *Advances in Neural Information Processing Systems*, 33:9900–9911, 2020.

Zhiyuan Chen and Bing Liu. *Lifelong machine learning*. Morgan et Claypool Publishers, 2018.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. December 2018.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of Single-Layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 215–223, Fort Lauderdale, FL, USA, 2011. PMLR.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. February 2017.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. October 2019.

Mahdi Milani Fard, Quentin Cormier, Kevin Canini, and Maya Gupta. Launch and iterate: Reducing prediction churn. *Advances in Neural Information Processing Systems*, 29, 2016.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *34th International Conference on Machine Learning, ICML 2017*, 3:2130–2143, June 2017.

Moritz Hardt, Eric Price, and Nathan Srebro. Equality of opportunity in supervised learning. October 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. December 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. 2015.

Mohammad Hossin and M.N Sulaiman. A review on evaluation metrics for data classification evaluations. *Int. J. Data Min. Knowl. Manag. Process*, 5(2):01–11, March 2015.

R Huang, A Geng, and Y Li. On the importance of gradients for detecting distributional shifts in the wild. *Thirty-Fifth Conference on Neural*, 2021.

Heinrich Jiang, Harikrishna Narasimhan, Dara Bahri, Andrew Cotter, and Afshin Rostamizadeh. Churn reduction via distillation. June 2021.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. U. S. A.*, 114(13):3521–3526, December 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`. Accessed: 2022-9-28.

Y Lecun, L Bottou, Y Bengio, and P Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, November 1998.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Front. Psychol.*, 4:504, August 2013.

Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? June 2019.

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

Geoff Pleiss, Tianyi Zhang, Ethan R Elenberg, and Kilian Q Weinberger. Identifying mislabeled data using the area under the margin ranking. *arXiv*, January 2020.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. March 2021.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, December 2019.

Kai Ming Ting and Ian H Witten. Stacked generalization: when does it work? `https://www.ijcai.org/Proceedings/97-2/Papers/011.pdf`, February 1997. Accessed: 2022-9-26.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2018.

Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, October 2021.

David H Wolpert. Stacked generalization. *Neural Netw.*, 5(2):241–259, January 1992.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. August 2017.

Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. Positive-Congruent training: Towards Regression-Free model updates. November 2020.

Ming Yin, Jennifer Wortman Vaughan, and Hanna Wallach. Understanding the effect of accuracy on trust in machine learning models. 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. October 2017.

Yue Zhao, Yantao Shen, Yuanjun Xiong, Shuo Yang, Wei Xia, Zhuowen Tu, Bernt Schiele, and Stefano Soatto. ELODI: Ensemble logit difference inhibition for Positive-Congruent training. May 2022.

7 Reproducibility

We will be releasing our GitHub repository containing all code required for precise replication of results up to differences in hardware. All experiments are implemented using a combination of PyTorch and PyTorch Lightning. We will also release access to the runs generated on Weights and Biases for easier analysis of our results without having to train models. Our code repository also contains Jupyter notebooks capable of generating all of the figures and tables in this paper.

APPENDIX

DESCRIPTION OF BASELINES EVALUATED

No Regularization f_{base} is learned on the original training data as follows

$$\theta_{\text{base}} = \arg\min_{\theta} \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} \ell(\phi(f(x;\theta)), y)$$

f_{new} is naively trained from a random initialization independently of f_{base} using the additional data $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{update}}$

$$\theta_{\text{new}} = \arg\min_{\theta} \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(\phi(f(x;\theta)), y)$$

This leads to an upper bound on churn that we aim to reduce.

Warm Start Instead of training f_{new} from a random initialization, it is initialized with the parameters of f_{base}. This reduces churn by biasing the parameters of f_{new} to be closer to f_{base}. This baseline is not always an option as on some datasets it results in worse accuracy than training from scratch using no regularization which is not a tradeoff we are willing to make. Ash & Adams (2020) have also observed that warm starts lead to worse generalization performance than retraining from scratch.

Distillation f_{new} is learned using a loss that is a combination of standard cross entropy loss on ground truth one-hot labels, and distillation using the predicted probabilities of f_{base} as targets

$$\theta_{\text{new}} = \arg\min_{\theta} \frac{1-\alpha}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(\phi(f(x;\theta)), y) + \frac{\alpha}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(\phi(f(x;\theta)), \phi(f(x;\theta_{\text{base}})))$$

where lower values of α place more emphasis on learning independently from $\mathcal{D}$, and higher values encourage matching the predictions of f_{base}.

Ensemble The base model is itself an ensemble $\mathcal{F}_{\text{base}} = \{f_{\text{base}}^{(1)}, ..., f_{\text{base}}^{(M)}\}$ where the parameters of each individual model $\theta_{\text{base}}^{(i)}, i \in [M]$ are learned as in the No Regularization baseline from different random initializations.

The new model is also an ensemble $\mathcal{F}_{\text{new}} = \{f_{\text{new}}^{(1)}, ..., f_{\text{new}}^{(M)}\}$ where the parameters of each individual model $\theta_{\text{new}}^{(i)}, i \in [M]$ are learned as in the No Regularization baseline from different random initializations.

Inference is done by averaging model logits via

$$\mathcal{F}(x) = \frac{1}{M} \sum_{i=1}^{M} f^{(i)}(x)$$

Anchor Loss Similar to distillation except that the distillation loss target depends on the correctness of the base model. f_{new} is learned using the following loss

$$\theta_{\text{new}} = \arg\min_{\theta} \frac{1-\alpha}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(\phi(f(x;\theta)), y) + \frac{\alpha}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \ell(\phi(f(x;\theta)), t(x,y))$$

$$t(x,y) = \begin{cases} \phi(f(x;\theta_{\text{base}})) & \text{if } \sigma(f(x;\theta_{\text{base}})) = y \\ \epsilon y & \text{otherwise} \end{cases}$$

Best Learner for Learned Model

Table 1 compares the effectiveness of logisic regression, random forest, and gradient boosting models as a trainable meta-model σ. Logistic regression does the best on all datasets except CIFAR100 suggesting there is no advantage to learning a non-linear function that combines the outputs of f_{new} and f_{base}. This observation is in accordance with the stacking literature where the meta-model is usually naive Bayes or logistic regression.

Table 1: Comparison logistic regression, random forest, and gradient boosting models for Learned Model stacking.

	Logistic Regression		Random Forest		Gradient Boosting	
Dataset	Acc $\uparrow$	$C_{\text{rel}} \downarrow$	Acc $\uparrow$	$C_{\text{rel}} \downarrow$	Acc $\uparrow$	$C_{\text{rel}} \downarrow$
MNIST	**99.374**	**0.162**	99.355	0.192	99.278	0.239
EMNIST	**94.391**	**1.095**	94.284	1.225	93.866	1.667
KMNIST	**96.359**	**0.679**	96.151	0.802	95.810	1.084
FashionMNIST	**91.760**	**1.573**	91.547	1.795	91.449	1.914
SVHN	**90.859**	**1.795**	90.457	2.033	90.163	2.241
CIFAR10	**84.839**	**2.618**	84.512	3.102	84.281	3.341
CIFAR100	46.067	11.714	**50.655**	**7.622**	40.694	15.310
STL10	**68.504**	**6.448**	67.349	7.005	65.855	8.411

Logistic Regression Hyperparameters We search over different values of the L2-regulariazation parameter $\lambda \in [1\mathrm{e}-4, 1\mathrm{e}-3, 1\mathrm{e}-2, 1\mathrm{e}-1]$

Random Forest Hyperparameters We consider random forests having $[50, 100, 250, 500]$ trees.

Gradient Boosting We consider $[50, 100, 250, 500]$ stages of gradient boosting.

Performance Boxplots

We visualize the spread of performance across runs using boxplots for both accuracy and churn to augment the results in our main table.

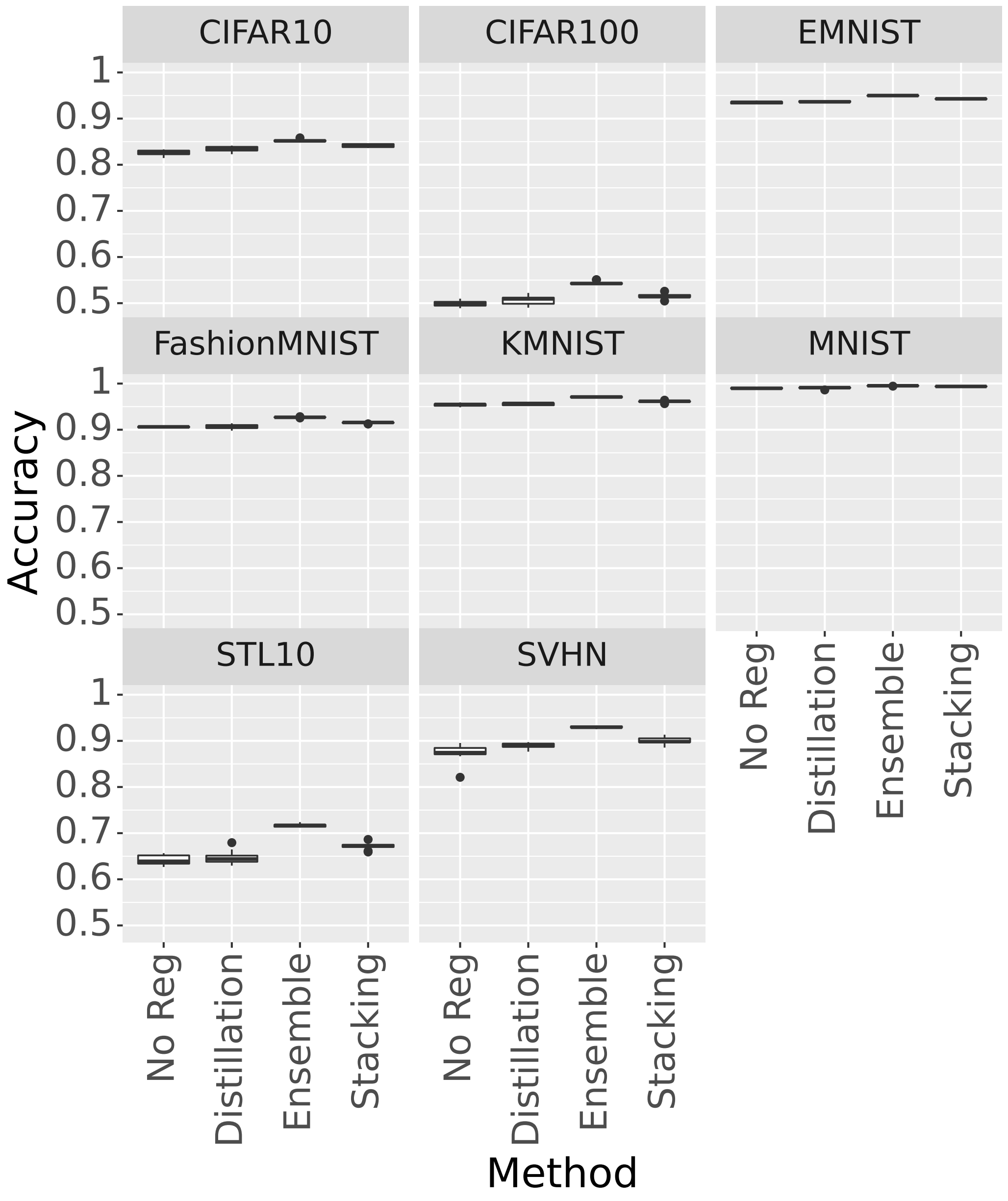


Figure A1: Accuracy of methods on all datasets. Ensemble gives the best accuracy as expected, albeit at a large computation cost.

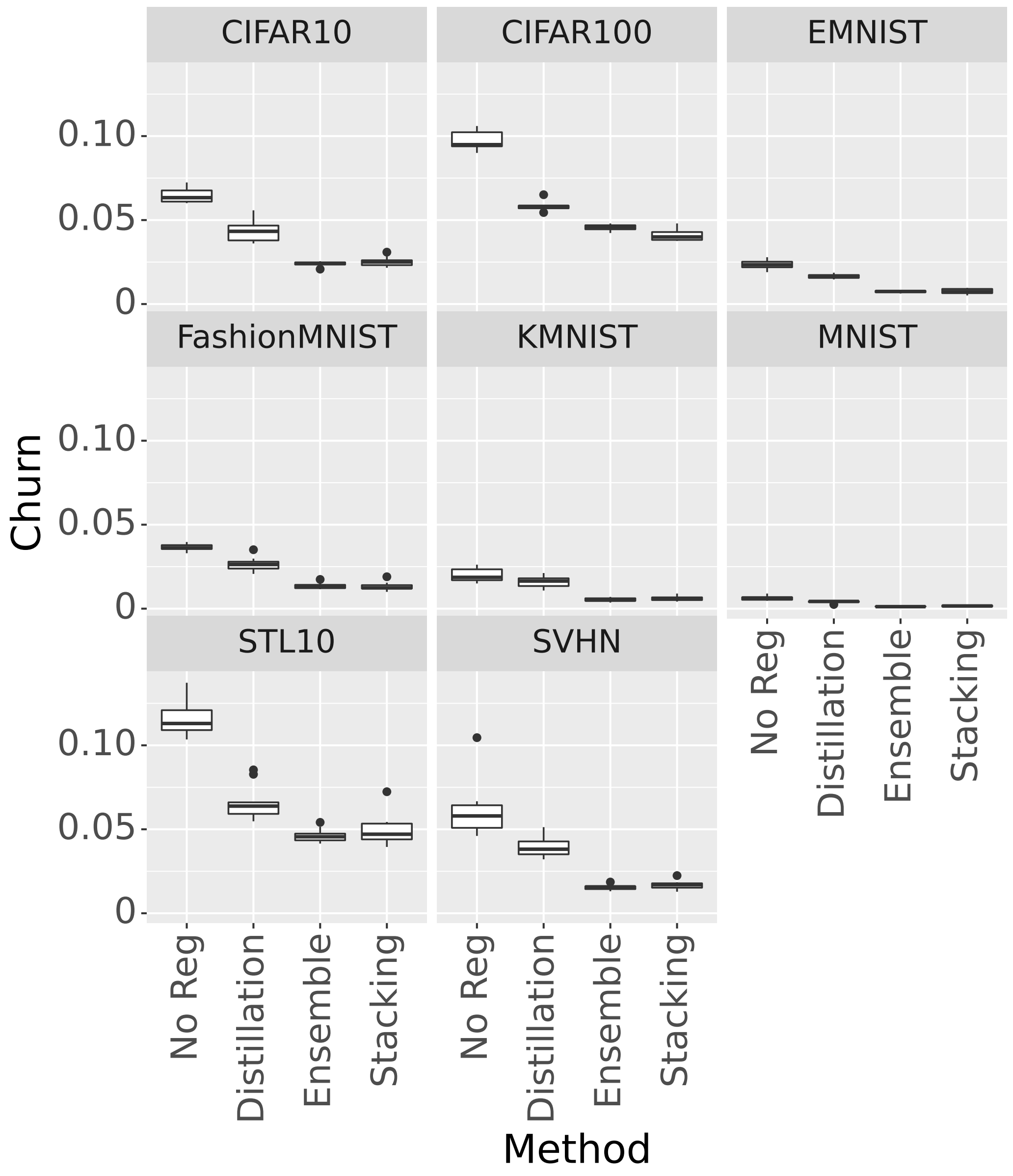


Figure A2: Churn of methods on all datasets. Ensemble and StackEm are comparable in churn reduction performance.

Prediction Confidence on Negative Flips

To get a better understanding of why calibrating models post-hoc with temperature scaling does not improve the effectiveness of the Confidence meta-model, we visualize the paired prediction confidence of f_{base} and f_{new} on negative flips where $\max_k f_{\mathrm{new}}(x)_k > \max_k f_{\mathrm{base}}(x)_k$ (Figure A3). If temperature scaling does not make f_{new} less confident on average, or f_{base} more confident on average, or both, these negative flips cannot be further reduced by a Confidence meta-model. Since both f_{base} and f_{new} are overconfident as was shown in the main text, temperature scaling does not change the ranking in prediction confidence on many samples, hence why negative flips are not reduced further. Moreover, even if f_{base} and f_{new} are perfectly calibrated, 0 negative flips cannot be achieved by the Confidence meta-model since if for example $\max_k f_{\mathrm{base}}(x)_k = 0.5$ and $\max_k f_{\mathrm{new}}(x)_k = 0.51$, then the probability of choosing the correct model is still a coin flip. Thus, full negative flip reduction would require occasionally choosing the lower confidence model.

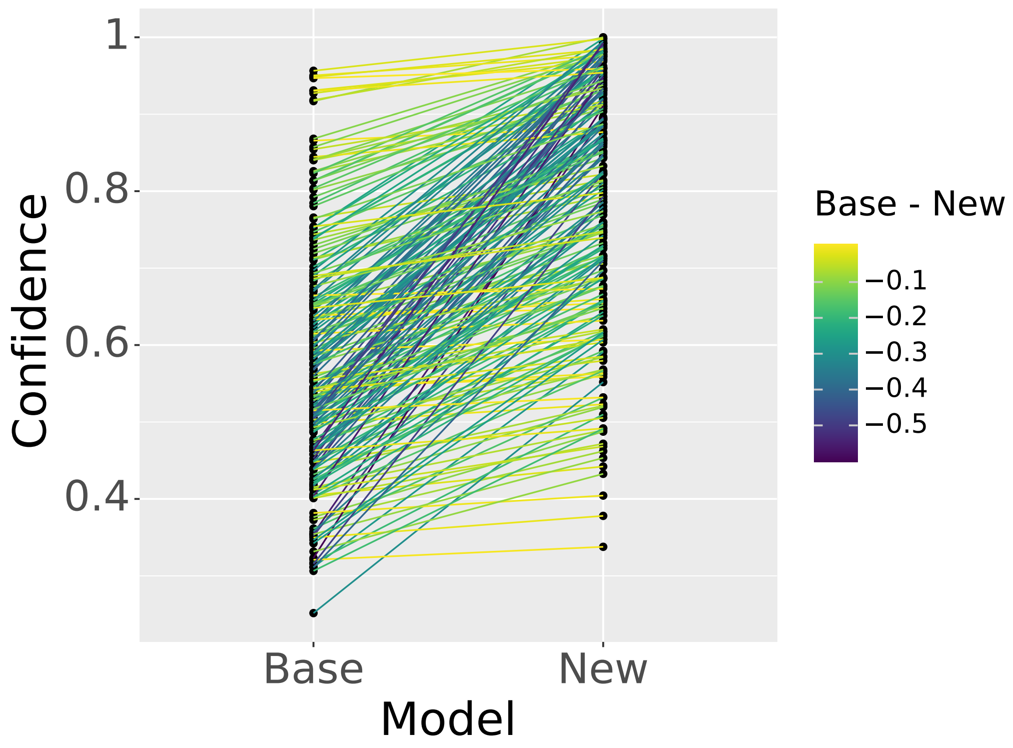


Figure A3: Paired prediction confidence of the predicted class for the base and new model on negative flips where the new model is more confident than the base model. The color indicates the difference in prediction confidence between the base and new model. ResNet18 model trained on CIFAR10 where the base model is trained on 30000 samples, and the new model is trained on an extra 10000 samples from scratch with no churn reduction regularization. These negative flips cannot be reduced further by a meta-model that chooses the model with highest prediction confidence since the new model predicts higher confidence than the base model but is incorrect.

Additional Baselines

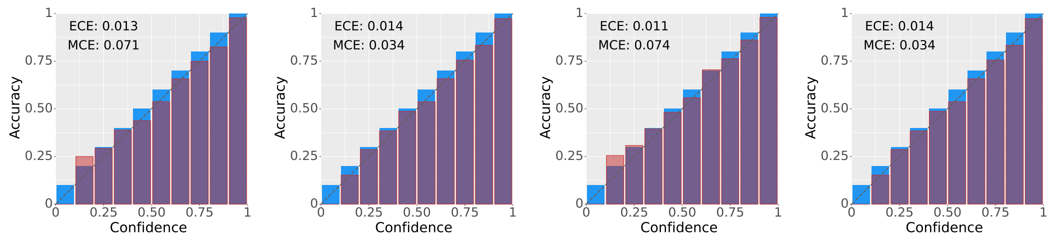
Table 2 shows results for 2 more baselines that did not fit in table 1. Note that results for the Warm Start baseline on FashionMNIST and CIFAR100 are not shown because Warm Start does not achieve the same accuracy as No Regularization, so churn reduction benefits are irrelevant in this case.

Table 2: Comparison of churn reduction methods on benchmark image classification datasets. StackEm Distillation outperforms both focal loss and warm start on both accuracy and C_{rel} (relevant churn) across all datatsets.

	No Regularization		Warm Start		Distillation		Anchor Loss	
Dataset	Acc $\uparrow$	$C_{\text{rel}} \downarrow$	Acc $\uparrow$	$C_{\text{rel}} \downarrow$	Acc $\uparrow$	$C_{\text{rel}} \downarrow$	Acc $\uparrow$	$C_{\text{rel}} \downarrow$
MNIST	98.958	0.640	99.028	0.552	99.072	0.410	99.162	0.419
EMNIST	93.490	2.359	93.505	2.109	93.647	1.656	93.818	1.734
KMNIST	95.405	1.991	95.499	1.665	95.585	1.595	95.832	1.594
FashionMNIST	90.601	3.642	Na	Na	90.653	2.634	91.044	2.724
SVHN	87.319	6.096	88.207	4.504	89.002	3.927	88.812	4.181
CIFAR10	82.598	6.450	82.792	5.406	83.372	4.354	83.383	4.681
CIFAR100	49.851	9.745	Na	Na	50.613	5.814	51.356	6.752
STL10	64.140	11.672	65.154	10.177	64.762	6.592	66.590	7.309

ADDITIONAL CALIBRATION FIGURES

SVHN



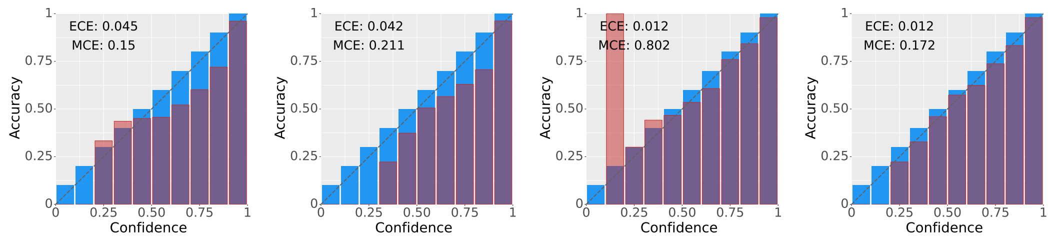
(a) f_{base} Uncalibrated (b) f_{new} Uncalibrated (c) f_{base} Calibrated (d) f_{new} Calibrated

Figure A4: Calibration of LeNet models trained on SVHN.

Table 3: Changes in prediction confidence ranking due to temperature scaling for LeNet models on SVHN.

Switch To	Benign	Good	Bad
f_{base}	0	0	0
f_{new}	1587	37	35

FASHIONMNIST



(a) f_{base} Uncalibrated (b) f_{new} Uncalibrated (c) f_{base} Calibrated (d) f_{new} Calibrated

Figure A5: Calibration of LeNet models trained on FashionMNIST.

Table 4: Changes in prediction confidence ranking due to temperature scaling for LeNet models on FashionMNIST.

Switch To	Benign	Good	Bad
f_{base}	601	2	3
f_{new}	118	12	10