FOCUS ON LIKELY CLASSES FOR TEST-TIME PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

We ask: ***Can focusing on likely classes of a single, in-domain sample improve accuracy? Prior work argued "no", we answer "yes" on average***. Standard entropy minimization yields largest gains, and we further dissect why by looking at its two core mechanisms: increasing likely and decreasing unlikely class predictions. Simply maximizing logits of the two most likely classes explains most of its benefits and, in some cases, even outperforms entropy minimization. Thus, decreasing classes is a secondary concern.

Our controlled experiment and theory investigate the most puzzling case, why simply optimizing the logits of the two most likely classes leads to accuracy gains. Our small scale networks suggest that the degree of uncertainty of a prediction, model choice, and to a lesser extent dataset choice moderate the success chances of the optimization. We show that the true class tends to have larger gradients independent of whether the prediction is correct or not. Our theory thoroughly investigates one of multiple possible explanations from the angle of shared features.

Our evaluation focuses on 26 large language models and 12 text datasets, and 18 image recognition models trained on ImageNet. It demonstrates gains of 0.6% on average for entropy minimization on uncertain samples using no extra data aside from the given test sample using a fixed learning rate. We also suggest *gradient-ray ensembling*. When traversing input/embedding space by applying the same gradient with differing learning rates and ***aggregating probabilities for the same sample further improves accuracy*** significantly to 0.75% relying on a coarse range for learning rates, thus improving accuracy and eliminating the need to find a precise learning rate at the expense of computation.

1 INTRODUCTION

State-of-the-art optimization methods in supervised and self-supervised learning, prevalent across subdisciplines ranging from image recognition in computer vision to text generation with large language models (LLMs), often minimize cross-entropy loss. The optimization objective aligns with the ideal scenario where a model assigns probability one to the correct class and zero to all others, achieving perfect class discrimination – even on random data (Zhang et al., 2021). However, during test-time, a model typically exhibits uncertainty, particularly in cases where it errs. Prior work showed that a prediction of a single sample can be improved using augmentations of that sample (e.g., MEMO Zhang et al. (2022)). However, finding the right augmentations can be non-trivial and evaluating many augmentations per single sample is computationally costly. Most works focus on domain adaption with the goal of continual learning based on many samples with a distribution shift. The shared dominant objective (e.g.,Wang et al. (2020); Hu et al. (2025a); Niu et al. (2023); Hu et al. (2025b)) ***entropy minimization was claimed to lead to a trivial solution for a single sample confirming just the most likely class (Wang et al., 2020)***. The claim seems reasonable as entropy minimization by definition just seems to make the most likely class more likely and even if it did not, it seems questionable that reducing entropy of just a single sample should fix errors more often than it causes them. In this work, we question the existing "wisdom" that optimizing a single sample leads to no gains. We provide compelling evidence against it. ***We also dissect entropy into two mechanisms: Increasing likely classes and decreasing unlikely classes***. The benefit of each is questionable: Classes beyond the most likely are known to provide valuable signals Hinton et al. (2015); Furlanello et al. (2018); Zhao et al. (2022). Thus, decrementing them might be harmful.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

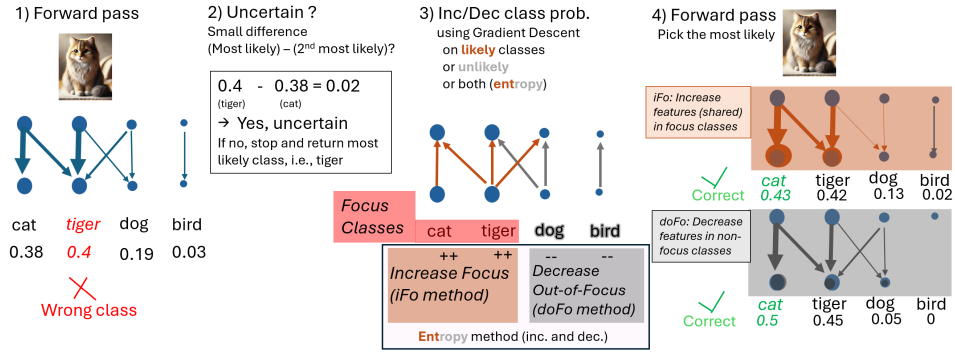


Figure 1: Test-time approaches (iFo/doFo/entropy): If a prediction exhibits high uncertainty, changes are done to focus on likely classes either by increasing focus classes or decreasing all others or both.

Also, the benefits of incrementing likely classes are unclear. Assume one increments the logits of the two most likely classes with the same learning rate. In typical test data (and as we also show) the most likely class is more often correct than the second most likely. So naively switching them just increases the error, also incrementing both with the same learning rate should just leave their order in place. Empirically, we show that each mechanism in isolation and their combination (entropy minimization) lead to gains on over 300 model-dataset pairs on average. Incrementing likely classes is clearly more beneficial than decrementing unlikely. To better understand why incrementing classes leads to gains (and why not) we perform a controlled experiment and conduct a theoretical analysis.

As the naive optimization method is computationally very expensive due to the need to conduct potentially tens of forward and backward passes per sample, we employ a number of computational optimizations, e.g., we do an uncertainty assessment step to limit our method to cases where it likely helps, we optimize only a small subset of parameters, i.e., token embeddings for text and we treat images as input parameters and optimize them. Still, our most computationally efficient method still requires one additional forward and backward pass to deliver accuracy gains of 0.6% and the identification of a model (but not dataset) dependent learning rate. However, we also identify a novel mechanism for self-ensembling, i.e. *gradient-ray ensembling*. It uses input/embedding space traversal that further improves accuracy by using more compute and also makes the identification of a learning rate mostly obsolete by allowing a coarse grained range. It aggregates probability estimates of classes obtained by applying the same gradient with different learning rates and picks the most likely class of the aggregate.

Our contribution is to provide fundamental insights into the workings of entropy optimization for test-time adaptation that are also of immediate practical value and a mechanism to trade computation for accuracy gains and engineering effort based on gradient-ray ensembling.

2 Methodology

Next, we describe how we aim to improve a prediction at test-time, as outlined in Figure 1. We add two components to the classical (single-forward) prediction process: (i) An *uncertainty assessment* procedure which decides whether to apply the focus optimization or not as gains are only probable on uncertain samples; (ii) A *focus optimization* module, which alters predictions.

Uncertainty Assessment: The prediction uncertainty assessment determines whether focus optimization should be applied to a specific sample or not. On certain samples our method has a low probability of improving predictions as we also discuss later. We quantify uncertainty based on how strongly a classifier favors the predicted class compared to other classes. We measure the uncertainty as the difference between the probabilities of the most likely and second-most likely classes. If this difference is large, the prediction is intuitively likely correct, as alternative classes have significantly lower probabilities. Concretely, we consider a model f, such as a neural network, that outputs logit values $f_c(X)$ for each class $c \in C$. The softmax layer converts the logit values $f_c(X)$ into probability estimates $p_c(X)$. Let $m(i)$ denote the class with the i-th highest probability for input X; thus, $p_{m(1)}(X)$ and $p_{m(2)}(X)$ represent the highest and second-highest probabilities, respectively.

We define the difference as:

$$\Delta_{1,2} = p_{m(1)}(X) - p_{m(2)}(X) \tag{1}$$

Similar separations have been made for domain adaptation but with a performance goal rather than to save on computation, e.g., Hu et al. (2025a).

Focus Optimization: We employ gradient descent for a fixed number of iterations (often just 1) for the methods illustrated in Figure 1: (i) *Increasing outputs of Focus classes F (iFo)*, (ii) *Decreasing outputs of Out-of-Focus classes $C \setminus F$ (doFo)* (C is the set of all classes), and (iii) doing both jointly (entropy). We do this for each sample anew (on the original network) as we require high learning rates that might negatively affect the network beyond the given sample. Visually, in the center panel of Figure 1, iFo increases activations of the subnetworks indicated by beige arrows, whereas doFo decreases those indicated by grey arrows. Entropy combines both ideas.

Entropy minimization implicitly defines focus classes $F \subset C$ to be increased and non-focus classes $C \setminus F$ to be decreased. Entropy is $L^{Ent}(X) = -\sum_{c \in C} p_c(X) \log p_c(X)$ (2), with softmax probabilities $p_c(X)$ defined as $p_c(X) = \frac{\exp(f_c(X))}{\sum_{k \in C} \exp(f_k(X))}$ (3).

The gradient with respect to logit $f_c(X)$ is $\frac{\partial L^{Ent}(X)}{\partial f_c(X)} = p_c(X) \left(\bar{f}(X) - f_c(X)\right)$ (4), where $\bar{f}(X)$ is the probability-weighted mean logit defined as $\bar{f}(X) = \sum_{k \in C} p_k(X) f_k(X)$ (5).

Using the chain rule, the entropy gradient with respect to the model parameters θ is therefore

$$\frac{\partial L^{Ent}(X)}{\partial \theta} = \sum_{c \in C} p_c(X) \left(\bar{f}(X) - f_c(X)\right) \frac{\partial f_c(X)}{\partial \theta}. \tag{6}$$

The term $p_c(X) \left(\bar{f}(X) - f_c(X)\right)$ acts as a scalar weight on the class-specific gradient $\frac{\partial f_c(X)}{\partial \theta}$. The weight is composed of two parts: the probability $p_c(X)$ regulates the scale of updates. In particular, without it unlikely classes with close to zero probability might receive substantial updates, which might hamper features. The second term has two functions: (i) it regulates scale by increasing updates for a class c the further its logit $f_c(X)$ is from the mean $\bar{f}(X)$ and (ii) its sign $\text{sign}\left(\bar{f}(X) - f_c(X)\right)$ implicitly defines whether a class belongs to the focus classes $c \in F$ or the non-focus classes. That is, $F^{Ent} = \left\{c \in C \mid f_c(X) > \bar{f}(X)\right\}$ (7). Applying $\log$ in Eq. 3 to obtain $\log p_c(X) = f_c(X) - \log \sum_k \exp(f_k(X))$, we can write $f_c(X) - \bar{f}(X) = \log p_c(X) + L^{Ent}(X)$ (8), and therefore $c \in F^{Ent} \iff p_c(X) > \exp\left(-L^{Ent}(X)\right)$ (9).

We also consider an explicit definition of F^k with $|F| = k = 2$, where F consists of the $k = 2$ most likely classes for a sample X. In this version, we drop the term $\left(\bar{f}(X) - f_c(X)\right)$, which implicitly defines F and, in our theoretical derivation, even $p_c(X)$. The motivation is two-fold: (i) entropy minimization gives zero gradient and thus has no impact in the most relevant obvious case of maximum uncertainty, e.g., if two classes are equally likely and all others have zero probability, i.e., $p_0 = p_1 = 0.5$; and (ii) this provides a simple version for understanding underlying mechanisms using a controlled experiment and theory.

Thus, we obtain the gradients for iFo and doFo with $F \in \{F^{Ent}, F^k\}$ based on Eq. 4:

$$\frac{\partial L^{iFo}(X)}{\partial \theta} = \sum_{c \in F} p_c(X) \left(\bar{f}(X) - f_c(X)\right) \frac{\partial f_c(X)}{\partial \theta}. \tag{10}$$

and

$$\frac{\partial L^{doFo}(X)}{\partial \theta} = \sum_{c \in C \setminus F} p_c(X) \left(\bar{f}(X) - f_c(X)\right) \frac{\partial f_c(X)}{\partial \theta}. \tag{11}$$

3 Experiments on real-world data

Setup. We evaluate our methods on more than 300 model-dataset pairs, i.e., vision and language models using ImageNet and diverse text datasets, respectively (see Appendix). We cover output classes ranging from about 10^3 to more than 10^6 and models range from below 0.1B to 27B. We use Algorithm 1 mostly with the stated defaults. However, we defined the uncertainty threshold $d_{1,2}$ implicitly. That is, we optimized a fixed percentile of $d_{1,2,\%}$ of all test samples, i.e., the 10% most uncertain text data meaning 30k samples per dataset (15% of images meaning 9k samples). This

Algorithm 1 Focus on the Likely: iFo, doFo, and Ent(ropy)

Require: $f(\cdot|\theta)$: Model with parameters θ; X: Input;
$Z^{Opt} \in \{X, \theta_{\text{emb}}, \theta\}$: Parameters to optimize (default: images: inputs X ; text: token embeddings θ_{emb})
$d_{1,2}$: Uncertainty threshold (default: 0.05, but mostly implicitly defined using $d_{1,2,\%}$;);
Opt $\in \{\text{Ent}, \text{iFo}, \text{iFo}^k, \text{doFo}\}$ (default: Ent);
(η,T): Initial learning rate η and Iterations T (default: Aggregating probabilities: text (1e-6, 13), images (1e-6/3,20); fixed lr: model-dependent (see Appendix);

Compute logits $f(X) = [f_c(X)]$ and probabilities $p_c^0(X) = \frac{e^{f_c(X)}}{\sum_{j \in C} e^{f_j(X)}}$ for $c \in C$
Sort probabilities so that $p^0_{m(1)}(X) \geq p^0_{m(2)}(X) \geq \dots$
Compute $\Delta_{1,2} = p^0_{m(1)}(X) - p^0_{m(2)}(X)$ {Eq. (1)}
if $\Delta_{1,2} \geq d_{1,2}$ **then** return **end if** {Uncertain Sample?}
if Opt $=$ iFo^k **then** $F \leftarrow \{m(i) \mid i \in [1,k]\}$ {Optimize only Top-k}
else {Optimize increment (dynamic) top-k or decrement non-top-k or both (=entropy)}
    Compute $L^{Ent}(X) = -\sum_{c \in C} p_c(X) \log p_c(X)$
    $F := \{c \in C \mid p_c(X) > \exp(-L^{Ent}(X))\}$ {Eq. (9)}
end if
$\nabla Z = \nabla_{Z^{Opt}} L^{Opt}(X)$ {Compute gradients for Z_{Opt} once}
for $t = 0$ to $T-1$ **do**
    $Z^{Opt} \leftarrow Z^{Opt} - \eta e^t \nabla Z$ {Update Z_{Opt}}
    Compute p_c^{t+1} using f with Z^{Opt} {Reevaluate f with changed parameters or inputs Z_{Opt}}
return $\arg\max_{c \in C} p_c^T(X)$ and $\arg\max_{c \in C}(\sum_i p_c^i(X))$ {Return most likely class for last iteration and aggregate}

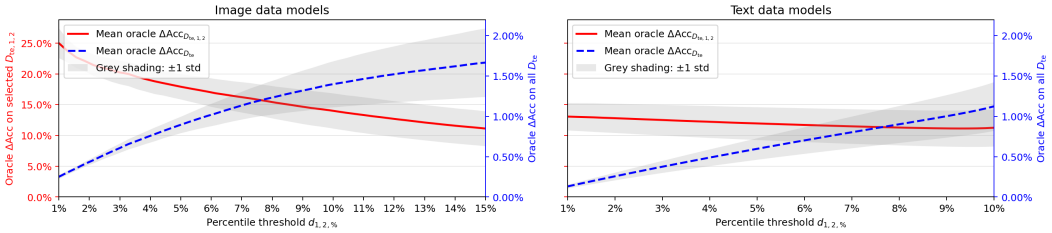


Figure 2: Larger threshold $d_{1,2,\%}$ implies more samples are uncertain, optimized and fixed so overall accuracy on all samples increases, but computational efficiency and gains on optimized, i.e., uncertain, samples decrease.

ensures that the number of uncertain samples is the same across models and datasets. The chosen percentage signifies a trade-off between computational effectiveness and overall accuracy gains as discussed in Experiment 1. We optimize using learning rates (lr) $1\text{e}{-6} \cdot 2^i$ for $i \in [0, 12]$ for text and pick the best one using a 50:50 validation/test split on a per model basis (for images we use $1/3\text{e}{-6}$ for $i \in [0, 20]$). These ranges are enough so that we essentially see no effect for our method for small lr due to Algorithm 1 as well as degradation for large lr as discussed in Experiment 2 and shown in Figure 3.

We compare the optimized model f^{our} with $our \in \{iFo, doFo, Ent\}$ or input X resulting from our optimization (Algorithm 1) against the original, unmodified model f or input X serving as baseline. We consider various configurations $O = O_{Img} \cup O_{Text} = \{f, D\}$, meaning pairs of models and datasets (f, D) for images O_{Img} and text O_{Text}. The test dataset D_{te} is split into sets of uncertain $D_{te,1,2}$ and certain samples. We report measures for individual configurations $(f, D) \in O$ on the entire held-out set, if no hyperparameter tuning took place, otherwise we use two-fold cross-validation. We commonly use Δ_{Acc} which typically measures accuracy between one of our methods and the baseline meaning the unmodified model and sample.

Experiment 1: Oracle gain and Uncertainty fraction: We vary the fraction of uncertain samples $d_{1,2,\%}$ and investigate the gain Δ_{Acc} on all test samples between oracle optimization and no opti-

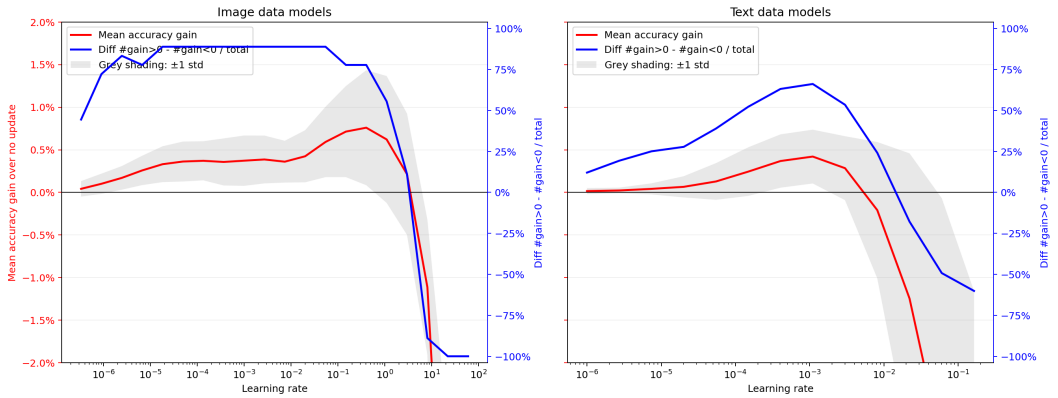


Figure 3: Average Δ_{Acc} for different learning rates

Table 1: Method comparison

Method	Δ_{Acc}	$\lvert O\rvert$ with Δ_{Acc} pos/neg	Larger/smaller Acc than Entropy pairs	Δ_{Acc} rel. to. Entropy
Entropy	$+0.60$ (± 0.75)	288/37	–	–
iFo	$+0.53$ (± 0.82)	277/48	117/206	-0.07 (± 0.74)
doFo	$+0.15$ (± 0.53)	207/117	65/260	-0.45 (± 0.74)
iFo^2	$+0.41$ (± 0.71)	261/61	99/225	-0.19 (± 0.67)

mization, i.e., an oracle can choose any of the evaluated learning rates for each sample to maximize accuracy. Thus, an oracle prediction is correct if applying just one learning rate yields the correct prediction but it cannot change gradient direction.

Results in Figure 2 show that oracle gains on uncertain samples ($D_{te,1,2}$, red line) can be as high as 25% for image data but drop with larger threshold $d_{1,2,\%}$. Larger threshold implies that also more certain samples are considered. Such samples are more likely to be correct, i.e., they cannot be fixed. Thus, the percentage gain decreases on the test data. In turn, this means that efficiency in terms of fixed samples per compute decreases. On all test data D_{te} the oracle gain continues to increase with larger threshold $d_{1,2,\%}$ but the growth gets slower - notably on image data. Also as shown in the next experiment even our best method's gain is more than an order of magnitude below oracle gain.

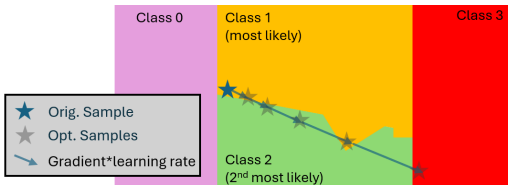
Experiment 2: Our accuracy gains

Figure 3 shows that for small learning rates, neither significant gains nor losses are observed on average and the number of classifiers that benefit from entropy optimization (positive accuracy gain) is about equal to those that do not. For very large learning rates performance decreases a lot due to optimization and becomes negative. In between we find that both the average gain and the number of models clearly favor optimization.

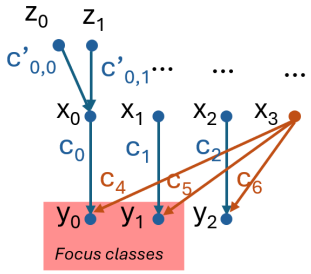
Table 1 compares the suggested methods for a fixed learning rate per model. Overall entropy performs best gaining on average 0.6% across all configurations though on about 10% of all configurations it worsens accuracy. The largest gain of entropy is due to incrementing likely classes (iFo). Just incrementing the top 2 classes (iFo^2) yields 2/3 of the entropy's gain, while decrementing less likely classes ($doFo$) leads only to a modest gain of 1/4 of entropy gains. Thus, the idea of removing "noise" by decrementing classes ($doFo$) is less beneficial than focusing on likely classes and simply increasing the top-2 classes. Interestingly, there are some models where iFo performs significantly better than entropy (e.g., Gemma), while also performing much worse (e.g., Olmo-2). We investigated several hypotheses related to model architecture and training regime but we could not conclusively factor out the decisive architectural difference. Thus, we opted for a controlled experiment as discussed in the next section.

Fixed learning rate (lr) vs gradient-ray ensembling *Experiment 2:* We computed accuracy using our proposed method *gradient-ray ensembling*, i.e., summing probabilities across all learning rates

and picking the most likely class (see Algorithm 1 and Figure 4a). For computational reasons, we summed probabilities for the initial top-16 classes and aggregated their probabilities. As shown in the Appendix, this yields much better outcomes than using just the top-4.[1]. For entropy we get $\Delta_{Acc} = 0.75(\pm 0.80)$ which is a statistically significant (p<0.001) increase compared to using a fixed lr. If we take only a subset of all evaluated learning rates to maximize accuracy (on a validation dataset) and then evaluate on the test data we get $\Delta_{Acc} = 0.84(\pm 0.79)$. More precisely, we started from all learning rates and performed leave-one-lr-out until the gain stopped improving. Gains for methods other than entropy are comparable (see Appendix).



(a) Movement of an uncertain sample when optimized towards few classes (e.g., class 1 and 2) with varying learning rates. It stays close to the decision boundary. Increasing common features tends to push it to the correct class.



(b) Theoretical model: 3 output classes y_i and one shared feature x_3 among all y_i

Figure 4: Illustrations of input movement and toy model.

4 Understanding Incrementing Classes

We aim to better understand why and how incrementing the logits of the top-2 classes for uncertain samples leads to accuracy gains. We restricted our analysis to the case where the correct class is among the top-2. We performed a controlled experiment that covers a number of basic architectures. We also discuss a more narrow theoretical scenario when accuracy gains can happen.

4.1 Controlled Experiment

We aim to better understand data and network conditions, in particular architecture effects when maximizing the top-2 classes using raw logits, i.e., we optimize inputs $\frac{\partial(f_{m(1)}(X)+f_{m(2)}(X))}{\partial X}$ (and also network parameters $\frac{\partial(f_{m(1)}(X)+f_{m(2)}(X))}{\partial \Theta}$). We train multiple small networks (below 200k parameters) covering fully connected networks, networks with different normalization layers (none, batchnorm, and layernorm) and w/o residual connections, and a transformer. We focused on character recognition based on EMNIST with 16 classes (0-9,A-F) and trained on roughly 1% of all data. This ensures that networks perform well ($> 90\%$ on test data) but there is still a large number of highly uncertain samples.

Our analysis reveals that the essential parameter predicting gains of incrementing top-2 classes is the uncertainty threshold $d_{1,2,\%}$. When focusing on highly uncertain samples, i.e., $d_{1,2,\%} = 0.02$, we consistently observe accuracy gains across all architectures when optimizing inputs X as well as all network parameters Θ. When focusing on samples that are in the 10% to 12% uncertainty percentile accuracy gains vanish when incrementing just raw unweighed logits. Using iFo that employs weighing of classes rather than just using raw logits helps for more certain samples. That is, weighing by probability p_c and logit distance to the mean $\overline{(f)}(X) - f_c(X)$ reduces the risk that a less likely (wrong) class overtakes a more likely (correct) class, but it also reduces the number of samples that are fixed through optimization.

For highly uncertain samples $d_{1,2,\%} = 0.02$, we also observe that the true class gradient magnitudes dominate those of wrong classes independent of their predicted rank. Assume the correct class is predicted wrongly, i.e., as second most likely. In this case, in more than 50% of all cases its gradient

[1]We expect using more classes to behave even better.

magnitude is still larger than that of the most likely class. The same holds if the true class is predicted correctly: Its gradient magnitude is largest among the top-2.

4.2 A POSSIBLE THEORETICAL RATIONALE

We investigate unweighed iFo^2, i.e., optimizing raw logits without probability weights, denoted as iFo^2_{noPr}.

We first prove for a simple model that iFo^2_{noPr} makes predictions rely more on shared, non-contrastive features. The example should be easy to follow for a reader, while being rigorous. Proofs and a generalized theory can be found in the Appendix. A feature x_i is shared between classes y_0 and y_1 if for a sample X changing the feature's value changes the logits of both classes. A non-contrastive feature for a set of classes C is a shared feature that either increases the logits of all classes C or decreases all of them. We show that iFo^2_{noPr} boosts shared features and investigate when it improves predictions. We show that iFo^2_{noPr} amplifies the noise-free part of the margin between the two likely classes while leaving the noise unchanged, i.e., it increases the signal-to-noise ratio of the prediction. The change in margin is network dependent, while for a shared, non-contrastive feature there is a wide range of network parameters where iFo^2_{noPr} helps. But for shared, contrastive features iFo^2_{noPr} can be hurtful.

Model: We assume three output classes $C = \{y_0, y_1, y_2\}$ and focus classes $F = \{y_0, y_1\}$. We have four input features $X = \{x_0, x_1, x_2, x_3\}$. For an illustration of our model, see Figure 4b. The input features x_j can be seen as originating from a prior layer l with parameters c' processing activations z, i.e., $x_i = l(z|c') = \sum_j c'_{j,i} z_j$, where z_j are activations for the current sample and $c'_{j,i}$ are trainable parameters. This corresponds to a pre-activation representation, or equivalently to a post-nonlinearity activation in a locally linear region (e.g., an active ReLU unit). We further assume some form of (layer) normalization, i.e., $\sum_j z_j^2 = 1$. To avoid a large number of symmetric case distinctions, we additionally consider a symmetric sample with $x_i = 1 \quad$ for all i. Outputs $y_i := f_i(X)$ are computed as:

$$y_0 = c_0 x_0 + c_4 x_3 \qquad y_1 = c_1 x_1 + c_5 x_3 \qquad y_2 = c_2 x_2 + c_6 x_3 \tag{12}$$

We apply the intuitive notion that a feature value cannot be negative, i.e., $x_i \geq 0$, which happens, for example, after layer activations pass through a non-linearity like $ReLU$. We also assume that $c_i \neq 0$ to avoid edge cases that can easily be deduced from our analysis. The model implicitly expresses that the presence of input features $\{x_0, x_1, x_2\}$ is only indicative of class i, i.e., a change of x_i alters only y_i for $i \in \{0, 1, 2\}$. In turn, $c_0 > 0$, $c_1 > 0$, and $c_2 > 0$. The shared feature x_3 impacts all outputs y_i. It might also be contrastive, i.e., it can be that two parameters from $\{c_4, c_5, c_6\}$ have opposing signs. Depending on the sign of c_4, c_5 and c_6 the presence of the feature ($x_3 > 0$) increases or decreases y_i. As we model dependence among classes only through the shared feature x_3, we assume that parameters $c'_{\cdot,i}$ are feature-specific, so that updating the parameters producing x_i does not directly affect x_m for $m \neq i$. This also facilitates our analysis.

Thus, we do not show any preference for a shared or non-shared feature and its upstream features z_i. We discuss deviating from this assumption after the proofs as they clarify the core behavior. We keep the head coefficients c_i fixed and update only the upstream parameters $c'_{j,i}$ (as our implementation tunes embeddings/inputs). We use the loss from our method iFo^2_{noPr} $L^{iFo^2_{noPr}} = -(y_0 + y_1)$ without weighting and the loss from optimizing just a single class $L_{y_i,-} = -y_i$. Since y_i is linear in x and x is linear in c', one gradient step changes the logits exactly. We write the learning rate as η. The theorems are stated for $\eta = 1$ and the proofs give the dependence on η. For an illustration of our model, see Figure 4b.

Theorem 1. *For a non-contrastive shared feature, the update to the shared feature is larger in magnitude for unweighed iFo^2_{noPr} than for the most likely class: $|c_4 + c_5| > \max(|c_4|, |c_5|)$. For a contrastive feature and most likely class y_0, the iFo^2_{noPr} update is smaller in magnitude than for the most likely class iff $|c_5| < 2|c_4|$. For a non-shared feature x_i of the most likely class y_i updates to x_i are identical.*

Next, we investigate when iFo^2_{noPr} improves predictions (and when it does not). Assume all parameters originating from training are perfect, so that without noise the predictions are always

correct. This can easily be achieved using sufficient training. Furthermore, if performance on training data is poor, there are chances that further training might improve also test-time performance - in particular, as overfitting is less of a concern in deep learning Belkin et al. (2019); Nakkiran et al. (2020). At test time, we assume the presence of noise, e.g., to account for differences between training and test data distribution.

As shown in the Preliminaries in the Appendix, the iFo^2_{noPr} update Δx_i does not depend on the values x_j but only on parameter values c_i, so additive noise on any feature passes through the update unchanged; in particular, we do not need to assume that the shared feature x_3 is noise-free. For concreteness, assume only x_0 is noisy (the cases of noise on x_1 or x_3 are analogous, see Remark 1), i.e.

$$\hat{x}_0 = x_0 + R_0, \qquad R_0 \sim \text{Uniform}[-1, 1].$$

Also, coefficients $c_0 = c_2 = c_5 = 1$. The coefficient c_4 for the shared feature is $c_4 = 1 + \delta_4$ and for the non-shared feature x_1 we have $c_1 = 1 + \delta_1$. Then

$$\hat{y}_0 = y_0 + c_0 R_0 = y_0 + R_0.$$

As we do not constrain δ_1, δ_4 to be either positive or negative, this setup covers many cases. δ_1 and δ_4 are assumed to arise from the training data. For example, consider binary features and a deterministic frequency-based linear classifier, where the coefficient of a feature is a monotone function of its frequency in the corresponding class (e.g., the log-odds under Bernoulli Naive Bayes). If the shared feature x_3 occurs slightly more often for class y_0 than the class-specific feature x_1 occurs for y_1, training yields $c_4 > c_1$, and hence $\delta_4 > \delta_1$ (and vice versa). Thus, a margin $m^* = \delta_4 - \delta_1$ (expressed later in Theorem 4) can be viewed as a small asymmetry learned from the training data, which iFo^2_{noPr} subsequently amplifies without amplifying the additive test noise.

It is also motivated by the fact that these assumptions lead to similar outputs y_i and thus high uncertainty if $|\delta_4|$ and $|\delta_1|$ are small. In the most extreme case, when $\delta_4 = \delta_1 = 0$ we get maximal uncertainty without noise $R_0 = 0$, meaning $y_0 = y_1$. To avoid this edge case, we assume $\delta_1 \neq \delta_4$. Note, that a key quantity in our analysis is $c_4 + c_5 = (1 + \delta_4) + 1$. We call a feature positively shared if $\delta_4 > -1$ so that $c_4 > 0$ and $c_5 > 0$ and, thus, $|c_4 + c_5| > \max(|c_4|, |c_5|)$. Let us focus on the situation, where the correct class is within the top-k classes and thus only analyze the outputs y_0, y_1. This is the most interesting case, as otherwise the prediction on the original classifier is already wrong (and thus changing the prediction in any way cannot cause a wrong prediction as it is already wrong, i.e., iFo^2_{noPr} might only help in terms of the 0/1 loss of this sample, although it may still reduce the margin of the correct class). For this, class y_2 must neither be among the two most likely classes before the step nor overtake after it, i.e., $y_2 < \min(y_0, y_1)$ and $y_2 + \Delta y_2 < \max(y_0 + \Delta y_0,\, y_1 + \Delta y_1)$. Since $\Delta y_2 = \eta c_6(c_4 + c_5) = \eta c_6(2 + \delta_4)$, while $\Delta y_0 = \eta(3 + 3\delta_4 + \delta_4^2) > 0$ and $\Delta y_1 = \eta\big(2 + \delta_4 + (1 + \delta_1)^2\big)$ (see the proof of Theorem 4), both conditions hold for all $R_0 \in [-1, 1]$ if, e.g., $c_6 < \min\{0, \delta_4, 1 + \delta_1\}$ and $\delta_4 > -2$.

Theorem 2 (iFo^2_{noPr} amplifies the noise-free margin)**.** *Let $m^* := \delta_4 - \delta_1 \neq 0$ be the noise-free gap $y_0 - y_1$, so that the correct class is y_0 if $m^* > 0$ and y_1 if $m^* < 0$, and let*

$$a := 1 + \eta(2 + \delta_4 + \delta_1).$$

One iFo^2_{noPr} step with learning rate η turns the gap $y_0 - y_1 = R_0 + m^*$ into

$$(y_0 + \Delta y_0) - (y_1 + \Delta y_1) = R_0 + a\, m^*.$$

The noise is unchanged and the noise-free gap is scaled by a. Hence iFo^2_{noPr} improves the margin of the correct class iff $a > 1$, i.e., iff $\delta_4 + \delta_1 > -2$; this holds in particular whenever x_3 is positively shared ($\delta_4 > -1$) and $c_1 > 0$ ($\delta_1 > -1$). If $\delta_4 + \delta_1 < -2$, the margin shrinks, and for $a < 0$ the noise-free ordering is reversed.

Corollary 1 (Accuracy)**.** *For $R_0 \sim \text{Uniform}[-1, 1]$, the probability of deciding correctly between y_0 and y_1 is $\frac{1}{2}\min\{2, \max\{0,\, 1 + a|m^*|\}\}$; before the step ($a = 1$) this is $\frac{1}{2}(1 + |m^*|)$ for $|m^*| \leq 1$, and it is non-decreasing in a.*

In terms of the original coefficients ($c_0 = c_5 = 1$): the noise-free gap is $c_4 - c_1$ and iFo^2_{noPr} adds $\eta(c_4^2 - c_1^2) = \eta(c_4 - c_1)(c_4 + c_1)$, so the condition $\delta_4 + \delta_1 > -2$ is simply $c_4 + c_1 > 0$. Thus the non-shared coefficient c_1 may be arbitrary as long as it is positive, which is no restriction for a feature with $x_1 > 0$ that is indicative of class y_1. To improve accuracy the toy experiment requires that the shared feature is more prevalent in the correct class. We conjecture: Accuracy improves by relying more on shared features as shared features are more prevalent in the correct class.

5 Related Work

To our knowledge, almost all prior works (see Table 2 in Appendix) have extra needs beyond the test sample. They require additional tasks (Liu et al., 2021; Sun et al., 2020) or data (potentially, self-generated using augmentations (Cotta, Memo, DeYO (Wang et al., 2022; Zhang et al., 2022; Lee et al., 2024)) or in-domain data (Liu et al., 2021) or in-domain samples(Hu et al., 2025b)). We intentionally focus on just a single test sample and refrain from such extra needs as they limit practical applicability, require effort, and often pose risks.Many studies on domain adaptation (see Table 2 in Appendix for a summary) and others (Kim et al., 2024; Osowiechi et al., 2024; Karmanov et al., 2024)) assume that test-time data originates from a domain different from that of the training data, whereas we focus on in-domain samples. Closest to our work in terms of problem setup is MemoZhang et al. (2022) as it also stresses robustness more broadly and it evaluates on in-domain samples. But its gains for in-domain samples rely on augmentations. Methodologically, many TTA methods are similar to one another and to ours (see Table 3 in Appendix). We differ by parameters optimized (we optimize inputs or their embeddings), loss functions (we dissect entropy) and, more interestingly, we propose a form of self-ensembling (gradient-ray ensembling). Some overlaps such as sample splitting appear coincidental as they serve different purposes or lead to opposing decisions. SAR optimizes only certain samples (measured by entropy), whereas we optimize the opposite, i.e., uncertain samples. PASLE, among other works in TTA, investigated focusing on more than one target class, e.g., hard labels for likely classes (Hu et al., 2025a) and entropy minimization (Wang et al., 2020; Zhang et al., 2022; Hu et al., 2025b). Commonly, test-time tuning relies on nearest neighbors and conducts a form of local learning by adjusting the model to the chosen data such as TSD+MLSC (Wang et al., 2023) and others (Sun et al., 2024; Bottou & Vapnik, 1992; Hardt & Sun, 2023; Hübotter et al., 2024). (Mummadi et al., 2021) aims at accounting for distribution shifts using the soft likelihood ratio (SLR) loss. Conceptually, iFo and doFo are multi-class optimizations, where we consider all focus (and out-of-focus) classes as target classes. Our technique is reflective as it often does not yield a decision based on a single forward pass (Zhong et al., 2024; Schneider, 2025; Madaan et al., 2024; Schneider & Vlachos, 2024; Selvaraju et al., 2017). When tuning embedding vectors, our technique can also be seen through the lens of moving the embedding vectors in a direction. This interpretation is well-known for word vectors with some evidence (Mikolov et al., 2013) for more abstract function vectors in LLMs (Todd et al., 2024). Further related work discussing "Test-time optimization of prompts and latent representations" and "Optimizing inputs instead of model parameters." is in the Appendix.

6 Limitations, Discussion and Conclusions

This work set out to answer the question *Does focusing on likely classes of a single, in-domain sample improve model predictions?* Applying entropy optimization yields "yes" on average, but gains are model, data, and uncertainty threshold dependent. Our study also hints that applying any test time adaptation technique on just say 3 networks and datasets has limited implications on its generality, as in our case it would be easy to pick a few datasets and models that make any of the methods iFo, iFo^2, doFo, and Entropy look much more favorable than they might appear across a large number of datasets and models. Still, despite using a large number of datasets and models the same criticism also applies to our study, i.e., applying even more datasets and more models might change outcomes. For example, while we included model families with different sizes, e.g., "Qwen/Qwen3.5-2B-Base" and its 9B version, and found no clear differences, very large models might alter our findings. Surprisingly simply increasing the two most likely classes captures most of the benefit of entropy minimization suggesting that gains are less due to reducing "unorderedness" as the term implies but rather due to lifting the right candidate classes. In particular, our controlled experiment showed that the gradient magnitude of the correct class tends to be larger. Our theory showed a set of conditions, when increasing the two most likely classes improves predictions, but other scenarios are likely plausible as well.

Furthermore by aggregating probabilities of classes for different learning rates (gradient-ray ensembling) gains can be further increased, which also is worth studying in more depth. We used softmax probabilities, for which calibration is a known issue(Guo et al., 2017) though less for pre-trained LLMs (Xie et al., 2024). Applying calibration techniques is future work.

AI USE STATEMENT

Yes, to aid or polish writing. AI (ChatGPT 5+6 and Claude Opus+Claude) was used for spelling and grammar checks.

Yes, for retrieval and discovery (e.g., finding related work). Most references were not found by AI, but a good number originate from AI. Related work was manually written except for 3 sentences in the appendix.

Yes, for research ideation or execution: A significant amount of code was AI-generated. The original algorithm was manually done, but later optimization and evaluation were mostly AI-generated, in particular the controlled experiment.

Yes, for proving mathematical claims. The toy example was mostly done manually, but refined by AI. The general theory in the Appendix included in the supplement submission was mostly AI-generated.

REFERENCES

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019. doi: 10.1073/pnas.1903070116.

Léon Bottou and Vladimir Vapnik. Local learning algorithms. *Neural computation*, 4(6):888–900, 1992.

Yining Chen, Colin Wei, Ananya Kumar, and Tengyu Ma. Self-training avoids using spurious features under domain shift. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://arxiv.org/abs/2006.10032.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020.

Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 1607–1616, 2018.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. *arXiv preprint arXiv:2305.18466*, 2023.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

Yihao Hu, Congyu Qiao, Xin Geng, and Ning Xu. Selective label enhancement learning for test-time adaptation. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*, 2025a.

Zixuan Hu, Yichun Hu, Xiaotong Li, Shixiang Tang, and Ling-Yu Duan. Beyond entropy: Region confidence proxy for wild test-time adaptation. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*, Vancouver, Canada, 2025b. URL https://arxiv.org/abs/2505.20704.

Jonas Hübotter, Sascha Bongni, Ido Hakimi, and Andreas Krause. Efficiently learning at test-time: Active fine-tuning of llms. *arXiv preprint arXiv:2410.08020*, 2024.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://arxiv.org/abs/1806.07572.

Adilbek Karmanov, Dayan Guan, Shijian Lu, Abdulmotaleb El Saddik, and Eric Xing. Efficient test-time adaptation of vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14162–14171, 2024.

Eungyeup Kim, Mingjie Sun, Christina Baek, Aditi Raghunathan, and J Zico Kolter. Test-time adaptation induces stronger accuracy and agreement-on-the-line. *Advances in Neural Information Processing Systems*, 37:120184–120220, 2024.

Jaehoon Lee, Daehyung Jung, Soobin Lee, Jinhwan Park, Jaehyuk Shin, Ui-Hyung Hwang, and Sungroh Yoon. Entropy is not enough for test-time adaptation: From the perspective of disentangled factors. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=XXXXX.

Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34:21808–21820, 2021.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5188–5196, 2015. doi: 10.1109/CVPR.2015.7299155.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR) Workshop*, 2013. URL https://arxiv.org/abs/1301.3781.

Chaithanya Kumar Mummadi, Robin Hutmacher, Kilian Rambach, Evgeny Levinkov, Thomas Brox, and Jan Hendrik Metzen. Test-time adaptation to distribution shift by confidence maximization and input transformation. *arXiv preprint arXiv:2106.14999*, 2021.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1g5sA4twr.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. *arXiv preprint arXiv:2302.12400*, 2023.

Philipp Oberdiek, Matthias Rottmann, and Hanno Gottschalk. Classification uncertainty of deep neural networks based on gradient information. In *Artificial Neural Networks in Pattern Recognition*, volume 11081 of *Lecture Notes in Computer Science*, pp. 113–125. Springer, 2018. doi: 10.1007/978-3-319-99978-4_9.

David Osowiechi, Gustavo A Vargas Hakim, Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Moslem Yazdanpanah, Ismail Ben Ayed, and Christian Desrosiers. Nc-ttt: A noise constrastive approach for test-time training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6078–6086, 2024.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. COLD decoding: Energy-based constrained text generation with langevin dynamics. In *Advances in Neural Information Processing Systems*, volume 35, pp. 9538–9551, 2022.

Johannes Schneider. Generative to agentic ai: Survey, conceptualization, and challenges. *arXiv preprint arXiv:2504.18875*, 2025.

Johannes Schneider and Michalis Vlachos. Reflective-net: Learning from explanations. *Data Mining and Knowledge Discovery*, 38(5):2975–2996, 2024.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 14274–14289, 2022.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations*, 2014.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. *arXiv preprint arXiv:2404.03646*, 2024. URL https://arxiv.org/abs/2404.03646.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Shuai Wang, Daoan Zhang, Zipei Yan, Jianguo Zhang, and Rui Li. Feature alignment and uniformity for test time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20050–20060, 2023.

Johnathan Xie, Annie S Chen, Yoonho Lee, Eric Mitchell, and Chelsea Finn. Calibrating language models with adaptive temperature scaling. *arXiv preprint arXiv:2409.19817*, 2024.

Wengao Ye, Yan Liang, and Lianlei Shan. Thinking on the fly: Test-time reasoning enhancement via latent thought policy optimization. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=r1WEQzkCQv.

Davis Yoshida and Kevin Gimpel. Reconsidering the past: Optimizing hidden states in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 4099–4105. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-emnlp.346.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. *Advances in neural information processing systems*, 35:38629–38642, 2022.

Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11953–11962, 2022. doi: 10.1109/CVPR52688.2022.01165.

Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu, Yanjun Lyu, Peng Shu, Xiaowei Yu, et al. Evaluation of openai o1: Opportunities and challenges of agi. *arXiv preprint arXiv:2409.18486*, 2024.

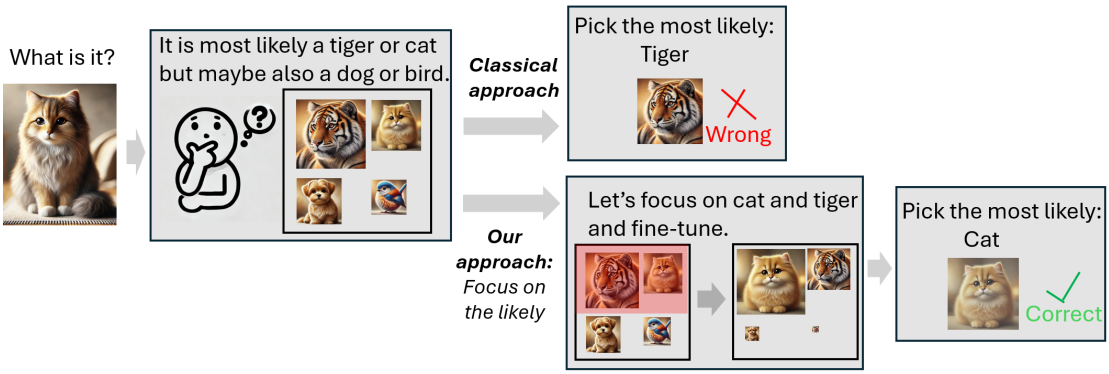


Figure 5: Overview: Fine-tune based on likely initial classes.

	In-Domain		1 Test Sample		No extras
Paper	Key topic	Eval	Per batch / Eval	Model Reset	
Cotta, TTT++,TSD+MLSC	×	×	×/×	×	×
TTT, ReCAP, DeYO	×	×	✓/✓	×	×
Memo	(✓)	(✓)	✓/×	×	×
Tent, PASLE	×	✓	✓/×	×	✓
SAR	×	×	✓/✓	×	✓
Ours	✓	✓	✓/✓	✓	✓

Table 2: Works on similar problems discussed in related work.

A THEORY FOR TOY EXAMPLE

Preliminaries: To study changes in the features, we consider one gradient step on the upstream parameters $c'_{j,i}$. Since $x_i = \sum_j c'_{j,i} z_j$, we have

$$\frac{\partial L}{\partial c'_{j,i}} = \frac{\partial L}{\partial x_i}\frac{\partial x_i}{\partial c'_{j,i}} = \frac{\partial L}{\partial x_i} z_j.$$

Hence, for learning rate η,

$$\Delta x_i = \sum_j \Delta c'_{j,i} z_j = -\eta \sum_j \frac{\partial L}{\partial c'_{j,i}} z_j = -\eta \frac{\partial L}{\partial x_i} \sum_j z_j^2 = -\eta \frac{\partial L}{\partial x_i}. \tag{13}$$

Note that $\partial L / \partial x_i$ depends only on the coefficients c_i and not on the feature values; hence Δx_i is the same for noisy and noise-free features. The feature-level derivatives are therefore

$$\frac{\partial L_{y_j,-}}{\partial x_i} = \begin{cases} -c_j, & i = j, \\ -c_{4+j}, & i = 3, \\ 0, & \text{otherwise}, \end{cases} \tag{14}$$

$$\frac{\partial L^{iFo^2_{noPr}}}{\partial x_i} = \begin{cases} -c_0, & i = 0, \\ -c_1, & i = 1, \\ -(c_4 + c_5), & i = 3, \\ 0, & \text{otherwise}. \end{cases} \tag{15}$$

Thus, for $\eta = 1$, the induced feature changes are

$$\Delta x_i^{y_j} = \begin{cases} c_j, & i = j, \\ c_{4+j}, & i = 3, \\ 0, & \text{otherwise}, \end{cases} \tag{16}$$

$$\Delta x_i^{iFo^2_{noPr}} = \begin{cases} c_0, & i = 0, \\ c_1, & i = 1, \\ c_4 + c_5, & i = 3, \\ 0, & \text{otherwise}. \end{cases} \tag{17}$$

Table 3: Methodological comparison with adaptation methods.

Aspect	PASLE	Tent	ReCAP	SAR	**Ours**
Loss Function	Custom	Entropy	(localized) Entropy	(reliable,flat) Entropy	Entropy, iFo, doFo, iFo^2
Parameters optimized	All	Norm. Layers	Norm. Layers	Norm. Layers	Inputs (or their embeddings)
Pseudo-labels	Hard, custom loss	-	-	-	Hard, soft weighting
Sample splitting	✓	×	×	✓	✓
Optimize only subset	×	×	×	✓(certain)	✓(uncertain)
No extra needs	✓	×(augment)	× (in-domain data)	✓	✓
Self-ensembling	×	×(augment)	× ×	×	✓(gradient-ray)

Theorem 3. *For a non-contrastive shared feature, the update to the shared feature is larger in magnitude for iFo^2_{noPr} than for the most likely class: $|c_4 + c_5| > \max(|c_4|, |c_5|)$. For a contrastive feature and most likely class y_0, the iFo^2_{noPr} update is smaller in magnitude than for the most likely class iff $|c_5| < 2|c_4|$. For a non-shared feature x_i of the most likely class y_i updates to x_i are identical.*

Proof. By definition, a non-contrastive feature x_3 either increases or decreases both logits, thus $\operatorname{sgn}(c_4) = \operatorname{sgn}(c_5)$. For the shared feature x_3 for iFo^2_{noPr} we have that its change is $\Delta x_3 = c_4 + c_5$ and for the most likely class it is either c_4 or c_5 depending on whether y_0 or y_1 is the most likely class. If a feature is non-contrastive then we have $\operatorname{sgn}(c_4) = \operatorname{sgn}(c_5)$ and thus $|c_4 + c_5| = |c_4| + |c_5| > \max(|c_4|, |c_5|)$, i.e., the update is larger for non-contrastive shared features for iFo^2_{noPr} ($k = 2$) than when updating the most likely class $k = 1$ (which can be either y_0 or y_1). If $\operatorname{sgn}(c_4) \neq \operatorname{sgn}(c_5)$, then $|c_4 + c_5| = \big||c_4| - |c_5|\big|$, which is smaller than $|c_4|$ iff $|c_5| < 2|c_4|$, and for $|c_5| > 2|c_4|$ we have $|c_4 + c_5| = |c_5| - |c_4| > |c_4|$ with $\operatorname{sgn}(c_4 + c_5) = \operatorname{sgn}(c_5)$. For non-shared features x_0, x_1 the update is $\Delta x_j = c_j$ for $j \in \{0, 1\}$ under both losses. □

Next, we investigate when iFo^2_{noPr} improves predictions (and when it does not). Assume all parameters originating from training are perfect, so that without noise the predictions are always correct. This can easily be achieved using sufficient training. Furthermore, if performance on training data is poor, there are chances that further training might improve also test-time performance - in particular, as overfitting is less of a concern in deep learning (Belkin et al. (2019); Nakkiran et al. (2020)). At test time, we assume the presence of noise, e.g., to account for differences between training and test data distribution.

As shown in the Preliminaries, the iFo^2_{noPr} update Δx_i does not depend on the feature values, so additive noise on any feature passes through the update unchanged; in particular, we do not need to assume that the shared feature x_3 is noise-free. For concreteness, assume only x_0 is noisy (the cases of noise on x_1 or x_3 are analogous, see Remark 1), i.e.

$$\hat{x}_0 = x_0 + R_0, \qquad R_0 \sim \text{Uniform}[-1, 1].$$

Also, coefficients $c_0 = c_2 = c_5 = 1$. The coefficient c_4 for the shared feature is $c_4 = 1 + \delta_4$ and for the non-shared feature x_1 we have $c_1 = 1 + \delta_1$. Then

$$\hat{y}_0 = y_0 + c_0 R_0 = y_0 + R_0.$$

As we do not constrain δ_1, δ_4 to be either positive or negative, this setup covers many cases. It is also motivated by the fact that these assumptions lead to similar outputs y_i and thus high uncertainty if $|\delta_4|$ and $|\delta_1|$ are small. In the most extreme case, when $\delta_4 = \delta_1 = 0$ we get maximal uncertainty without noise $R_0 = 0$, meaning $y_0 = y_1$. To avoid this edge case, we assume $\delta_1 \neq \delta_4$. Note, that a key quantity in our analysis is $c_4 + c_5 = (1 + \delta_4) + 1$. We call a feature positively shared if $\delta_4 > -1$ so that $c_4 > 0$ and $c_5 > 0$ and, thus, $|c_4 + c_5| > \max(|c_4|, |c_5|)$. Let us focus on the situation, where the correct class is within the top-k classes and thus only analyze the outputs y_0, y_1. This is the most interesting case, as otherwise the prediction on the original classifier is already wrong (and thus changing the prediction in any way cannot cause a wrong prediction as it is already wrong, i.e., iFo^2_{noPr} might only help in terms of the 0/1 loss of this sample, although it may still reduce the margin of the correct class). For this, class y_2 must neither be among the two most likely classes before the step nor overtake after it, i.e., $y_2 < \min(y_0, y_1)$ and $y_2 + \Delta y_2 < \max(y_0 + \Delta y_0,\ y_1 + \Delta y_1)$. Since $\Delta y_2 = \eta c_6(c_4 + c_5) = \eta c_6(2 + \delta_4)$, while $\Delta y_0 = \eta(3 + 3\delta_4 + \delta_4^2) > 0$ and $\Delta y_1 = \eta\big(2 + \delta_4 + (1 + \delta_1)^2\big)$ (see the proof of Theorem 4), both conditions hold for all $R_0 \in [-1, 1]$ if, e.g., $c_6 < \min\{0, \delta_4, 1 + \delta_1\}$ and $\delta_4 > -2$.

Theorem 4 (iFo^2_{noPr} amplifies the noise-free margin)**.** *Let $m^* := \delta_4 - \delta_1 \neq 0$ be the noise-free gap $y_0 - y_1$, so that the correct class is y_0 if $m^* > 0$ and y_1 if $m^* < 0$, and let*

$$a := 1 + \eta(2 + \delta_4 + \delta_1).$$

One iFo^2_{noPr} step with learning rate η turns the gap $y_0 - y_1 = R_0 + m^*$ into

$$(y_0 + \Delta y_0) - (y_1 + \Delta y_1) = R_0 + a\, m^*.$$

The noise is unchanged and the noise-free gap is scaled by a. Hence iFo^2_{noPr} improves the margin of the correct class iff $a > 1$, i.e., iff $\delta_4 + \delta_1 > -2$; this holds in particular whenever x_3 is positively shared ($\delta_4 > -1$) and $c_1 > 0$ ($\delta_1 > -1$). If $\delta_4 + \delta_1 < -2$, the margin shrinks, and for $a < 0$ the noise-free ordering is reversed.

Corollary 2 (Accuracy)**.** *For $R_0 \sim \mathrm{Uniform}[-1, 1]$, the probability of deciding correctly between y_0 and y_1 is $\frac{1}{2}\min\{2, \max\{0,\, 1 + a|m^*|\}\}$; before the step ($a = 1$) this is $\frac{1}{2}(1 + |m^*|)$ for $|m^*| \leq 1$, and it is non-decreasing in a.*

In terms of the original coefficients ($c_0 = c_5 = 1$): the noise-free gap is $c_4 - c_1$ and iFo^2_{noPr} adds $\eta(c_4^2 - c_1^2) = \eta(c_4 - c_1)(c_4 + c_1)$, so the condition $\delta_4 + \delta_1 > -2$ is simply $c_4 + c_1 > 0$. Thus the non-shared coefficient c_1 may be arbitrary as long as it is positive, which is no restriction for a feature with $x_1 > 0$ that is indicative of class y_1.

Proof. By the Preliminaries, $\Delta x_0 = \eta c_0$, $\Delta x_1 = \eta c_1$, $\Delta x_3 = \eta(c_4 + c_5)$, independently of R_0. Since the head is fixed, $\Delta y_0 = c_0 \Delta x_0 + c_4 \Delta x_3 = \eta\big(c_0^2 + c_4(c_4 + c_5)\big)$ and $\Delta y_1 = \eta\big(c_1^2 + c_5(c_4 + c_5)\big)$, hence

$$\Delta y_0 - \Delta y_1 = \eta\big[(c_0^2 - c_1^2) + (c_4^2 - c_5^2)\big] = \eta\big[\delta_4(2 + \delta_4) - \delta_1(2 + \delta_1)\big] = \eta(\delta_4 - \delta_1)(2 + \delta_4 + \delta_1) = (a - 1)\, m^*.$$

Adding the original gap $R_0 + m^*$ gives $R_0 + a\, m^*$. Since a is increasing in $\delta_4 + \delta_1$ with $a = 1$ at $\delta_4 + \delta_1 = -2$ and $a = 0$ at $\delta_4 + \delta_1 = -2 - 1/\eta$, the remaining claims follow; $\delta_4 > -1$ and $\delta_1 > -1$ imply $\delta_4 + \delta_1 > -2$. For the corollary, if $m^* > 0$ the decision is correct iff $R_0 > -a\, m^*$, and $\Pr(R_0 > -t) = \frac{1}{2}\min\{2, \max\{0, 1 + t\}\}$ for $R_0 \sim \mathrm{Uniform}[-1, 1]$; the case $m^* < 0$ is symmetric. □

Remark 1: For $\delta_4 = \delta_1$ (in particular for perfect symmetry $\delta_4 = \delta_1 = 0$), iFo^2_{noPr} has no impact, i.e., Before iFo^2_{noPr},

$$y_0 - y_1 = R_0.$$

After iFo^2_{noPr},

$$(y_0 + \Delta y_0) - (y_1 + \Delta y_1) = R_0.$$

Thus, the ground truth assigns equal likelihood to both classes, and the outcome depends only on noise. In this case, iFo^2_{noPr} has no effect. Remark 2: Deviating from a symmetric setup and using $x_3 \neq 1$ (with $x_0 = x_1 = 1$) does not change the update, because Δx_i does not depend on the feature values: the gap after the step is

$$R_0 + \delta_4 x_3 - \delta_1 + \eta(\delta_4 - \delta_1)(2 + \delta_4 + \delta_1).$$

The feature value x_3 only enters the noise-free gap $\delta_4 x_3 - \delta_1$. For $\delta_1 = 0$ the gap is $R_0 + \delta_4\big(x_3 + \eta(2 + \delta_4)\big)$: iFo^2_{noPr} improves the margin iff $\delta_4 > -2$, independently of x_3, and a larger x_3 only moves the threshold for reversing the noise-free ordering from $\delta_4 > -2 - 1/\eta$ to $\delta_4 > -2 - x_3/\eta$. For $\delta_1 \neq 0$ iFo^2_{noPr} improves the margin iff $\mathrm{sgn}(\delta_4 x_3 - \delta_1) = \mathrm{sgn}\big((\delta_4 - \delta_1)(2 + \delta_4 + \delta_1)\big)$, which a larger x_3 can either help or hurt (e.g., $\delta_4 = -1.2$, $\delta_1 = -0.9$: the condition holds for $x_3 = \frac{1}{2}$ but not for $x_3 \in \{1, 2\}$). Preferring the shared feature in the update (a larger $\sum_j z_j^2$ for the parameters producing x_3, say $g > 1$) multiplies the shared part of the gap change, giving $\eta\big[-\delta_1(2 + \delta_1) + g\,\delta_4(2 + \delta_4)\big]$ instead of $\eta(\delta_4 - \delta_1)(2 + \delta_4 + \delta_1)$; for $\delta_1 = 0$ this changes the size but not the sign condition $\delta_4 > -2$.

Remark 1 (Noise on other features)**.** *If instead x_1 or x_3 is noisy, $\hat{x}_1 = 1 + R_1$ or $\hat{x}_3 = 1 + R_3$, the gap before the step is $R' + \delta_4 - \delta_1$ with $R' = -c_1 R_1 = -(1 + \delta_1)R_1$, respectively $R' = (c_4 - c_5)R_3 = \delta_4 R_3$, and Theorem 4 holds with R_0 replaced by R' (and Corollary 2 with the distribution of R'), because the update does not depend on the feature values.*

Remark 2 (Probability-weighted loss)**.** *With the weights $\tilde{p}_0, \tilde{p}_1$ of the main paper the change of the gap is*

$$\frac{\eta}{2}\Big[(\tilde{p}_0+\tilde{p}_1)(\delta_4-\delta_1)(2+\delta_4+\delta_1)+(\tilde{p}_0-\tilde{p}_1)\big(1+(1+\delta_1)^2+\delta_4^2\big)\Big].$$

For $\tilde{p}_0=\tilde{p}_1$ (the uncertain case) Theorem 4 holds with η replaced by $\eta\tilde{p}_0$. For $\tilde{p}_0\neq\tilde{p}_1$ the second term has the sign of $\tilde{p}_0-\tilde{p}_1$, i.e., of the current noisy gap R_0+m^*, and favors the currently leading class; it vanishes at a tie.

Remark 3 (Updating the head as well)**.** *If the head coefficients are updated too ($\Delta c_0=\eta\hat{x}_0=\eta(1+R_0)$, $\Delta c_1=\eta x_1$, $\Delta c_4=\Delta c_5=\eta x_3$), the head update adds $\eta(\hat{x}_0^2-x_1^2)=\eta(2R_0+R_0^2)$ to the gap, which is pure noise since $x_0=x_1=1$ without noise, and the products of head and feature updates add $\eta^2(R_0-\delta_1)$. Grouping the exact gap after the step into its noise-free and noise parts,*

$$(y_0+\Delta y_0)-(y_1+\Delta y_1)=\big[m^*\big(1+\eta(2+\delta_4+\delta_1)\big)-\eta^2\delta_1\big]+\big[(1+\eta)^2R_0+\eta R_0^2\big].$$

Compared with Theorem 4, the noise-free gap is amplified by the same factor (up to $-\eta^2\delta_1$), but the noise is amplified as well: its coefficient grows from 1 to $(1+\eta)^2$ and it acquires the bias $E[\eta R_0^2]=\eta/3$ in favor of the class with the noisy feature. For $\eta=1$ and $R_0\sim\mathrm{Uniform}[-1,1]$ the standard deviation of the noise grows by the factor $\sqrt{16/3+4/45}/\sqrt{1/3}\approx 4.0$ while the noise-free gap grows by $3+\delta_4+\delta_1-\delta_1/m^*$, so the signal-to-noise ratio improves only for $\delta_4+\delta_1\gtrsim 1$ (for $\delta_1=0$: $\delta_4>1.03$) instead of $\delta_4+\delta_1>-2$. This is why we keep the head fixed.

B Further related work

Test-time optimization of prompts and latent representations. Several works are particularly related in that they keep the pretrained model fixed and optimize continuous representations at inference time. Test-Time Prompt Tuning (TPT) learns continuous prompt embeddings from a single unlabeled test example by entropy minimization (Shu et al., 2022). Its entropy objective is therefore closely related to ours, but TPT obtains its self-supervision from confidence-selected augmentations and optimizes auxiliary prompt tokens rather than the representation of the observed input itself. In language models, Plug-and-Play Language Models (PPLM) use gradients from an attribute objective to modify hidden activations during generation while keeping the LM parameters fixed (Dathathri et al., 2020). Hidden-State Optimization (HSO) similarly performs gradient-based inference-time updates of cached hidden states rather than model parameters (Yoshida & Gimpel, 2021); its gradient, however, is obtained from the likelihood of observed evaluation tokens rather than solely from the model's current uncertain prediction. COLD decoding optimizes continuous relaxed token representations by gradient-based sampling under externally specified energy functions (Qin et al., 2022). These approaches establish inference-time optimization in continuous representation space, but use either auxiliary control objectives, observed tokens, augmentations, or generation constraints.

Recent work has moved this idea toward self-guided latent optimization in LLMs. Latent Thought Policy Optimization (LTPO) treats intermediate latent thought vectors as per-instance optimization variables while freezing all model parameters and uses an intrinsic confidence-based reward computed from the LLM's own output distribution (Ye et al., 2026). This is close to our single-sample, self-guided setting, but LTPO optimizes additional intermediate reasoning latents using policy gradients, whereas we directly modify the given input—pixels for vision models and the token embeddings of the observed context for language models—using gradients derived from the likely output classes.

Optimizing inputs instead of model parameters. Optimizing the input of a fixed neural network has a long history, although mostly for objectives different from improving ordinary test-time prediction. Simonyan et al. (2014) optimize image pixels to maximize a class score in order to visualize what a fixed classifier has learned. Mahendran & Vedaldi (2015) recover images by optimizing pixels so that a fixed network produces a target internal representation. Adversarial-example methods likewise optimize perturbations of the input while keeping the model fixed, but deliberately seek changes that induce prediction errors (Szegedy et al., 2014). These works establish input optimization as a general mechanism for interrogating or controlling fixed neural networks. Our use is different: the input update is performed on the given test sample itself, without a target label, with the aim of improving its prediction using only the model's own likely classes. Relatedly, Mummadi et al. (2021) learn an input transformation at test time for distribution-shift adaptation;

unlike our method, the optimized object is a transformation module rather than the current sample representation itself.

Other connections. Our more general theory relates to Neural Tangent Kernel theory Jacot et al. (2018). Our input-gradient matrix $K^X = J_X J_X^\top$ is an analogous Jacobian Gram matrix, whereas K^Θ is the same-input, output-by-output block of the empirical neural tangent kernel. Applying the top-two objective to either matrix gives the local margin velocity $K_{aa} - K_{bb}$, making the comparison between input and parameter optimization mathematically explicit. The tendency of that diagonal difference to favor the true class is an empirical property rather than a consequence of NTK theory.

Gradient norms have been evaluated in multiple applications, e.g., to assess uncertainty of predictions Oberdiek et al. (2018).

Chen et al. (2020) showed that entropy minimization can reduce reliance on spurious features. We said that doFo reduces spurious classes, which likely rely on spurious features.