Deep Learning on a Data Diet: Finding Important Examples Early in Training

-Summary-

Introduction - Preliminaries

Questions :

- How much of the data is superfluous?
- Which examples are important for generalization, and how do we find them?

=> Simple scores averaged over several weight initializations can be used to identify important examples very early in training.

Notation

- Training set : $S = \{(x_i, y_i)\}_{i=1}^{N}$ / $(x_i, y_i) \sim \mathcal{D}$ / $x_i \in \mathbb{R}^d$, $y_i \in \{0,1\}^K$ (one-hot vector)
- Logit output : $f_w(x) \in \mathbb{R}^K$, where $w \in \mathcal{W} \subseteq \mathbb{R}^D$ is weight for NN
- Softmax function : $\sigma(z_1, \ldots z_K)_k = \frac{\exp\{z_k\}}{\sum_{k'=1}^{K} \exp\{z_{k'}\}}$
- NN probability output : $p(w, x) = \sigma(f_w(x))$
- Cross-entropy loss : $\ell(p, y) = \sum_{k=1}^{K} y^{(k)} \log(p^{(k)})$, where p = NN probability output
- Iterates of SGD : $w_0, \ldots, w_T$ satisfying $w_t = w_{t-1} - \eta \sum_{(x,y) \in S_{t-1}} g_{(t-1)}(x, y)$

  ($S_0, \ldots, S_{T-1} \in S$ = sequence of size-M minibatches / $g_{t-1}(x, y) = \nabla_{w_{t-1}} \ell(p(w_{t-1}, x), y)$ for $t = 1, \ldots, T$

Gradient Norm Score and infinitesimal analysis

Definition : Gradient Normed score (GraNd score)

The **GraNd score** of a training example (x, y) at time t is :

$$\chi_t(x, y) = \mathbb{E}_{w_t}[\|g_t(x, y)\|_2]$$

Recall : $g_t(x, y) = \nabla_{w_t} \ell(p(w_t, x), y)$ / $p(w, x) = \sigma(f_w(x))$ / w_t is R.V due to random initialization of SGD

Intuition for GraNd score

Consider continuous time (i.e t is continuous / not discrete) for approximation

- Time derivative of the loss for (x, y) with gradient computed on the minibatch S_t :

$$\Delta_t\big((x, y), S_t\big) = -\frac{d\ell(p(w_t, x), y)}{dt} = \nabla_{w_t} \ell(p(w_t, x), y) \frac{dw_t}{dt} = g_t(x, y) \frac{dw_t}{dt} \text{ (by chain rule)}$$

- Discrete time dynamic approximation :

$$\frac{dw_t}{dt} \approx w_{t+1} - w_t = -\eta \sum_{(x,y) \in S_t} g_t(x, y)$$

Gradient Norm Score and infinitesimal analysis

Lemma 2.2

Let $S_{-j} = S - \{(x_j, y_j)\}$. Then for all (x^*, y^*), there exists c such that

$$|\Delta_t\big((x^*, y^*), S\big) - \Delta_t\Big((x^*, y^*), S_{-j}\Big)| \le c\|g_t\big(x_j, y_j\big)\|$$

Note : Here, we deal with S, not the minibatch S_t, but note that $S_t \subseteq S$

- If $\big(x_j, y_j\big) \notin S_t$, then $\Delta_t\big((x^*, y^*), S\big) = \Delta_t\Big((x^*, y^*), S_{-j}\Big)$ (no change)
- If $\big(x_j, y_j\big) \in S_t$, then $\Delta_t\big((x^*, y^*), S\big) - \Delta_t\Big((x^*, y^*), S_{-j}\Big) = \eta \cdot g_t(x^*, y^*)^T g_t\big(x_j, y_j\big) \le \eta\|g_t(x^*, y^*)\|\big\|g_t\big(x_j, y_j\big)\big\|$

Remark : (If $\big(x_j, y_j\big) \in S_t$)

By GD update rule, w_t induced by S = w_t induced by S_{-j} (start to differ from w_{t+1})

Gradient Norm Score and infinitesimal analysis

Interpretation of Lemma 2.2

- Expected value of $\|g_t(x_j, y_j)\|$ is exactly the GraNd score of (x_j, y_j)
- Small GraNd score in expectation have a bounded influence on learning how to classify the rest of the training data at a given training time
- Note : The converse does not hold :=> examples with large scores may have gradients that cancel out and do not contribute much (i.e : the upper bound is loose)

Approximation of GraNd

- Observe GraNd score $\chi_t(x, y) = \mathbb{E}[\left\|\sum_{k=1}^{K} \frac{d\ell(p(w_t,x),y)}{df_{w_t}^{(k)}(x)} \nabla_{w_t} f_{w_t}^{(k)}(x)\right\|_2]$ (Reformulation)
- Under Cross-Entropy loss, $\frac{d\ell(p(w_t,x),y)}{df_{w_t}^{(k)}(x)} = p(w_t, x)^{(k)} - y^{(k)}$
- From S.Fort (2019), logit gradients(= $\nabla_{w_t} f_{w_t}^{(k)}(x)$) are nearly orthogonal among classes throughout training in expectation.

Error L2-Norm score and forgetting score

Def : Error L2-Norm (EL2N) score

The **EL2N score** of a training sample (x, y) is defined to be :

$$\mathbb{E}[\|p(w_t, x) - y\|_2]$$

Definition : forgetting event / forgetting score (~ Toneva et al.,2018)

- **Forgetting event of** $(\mathbf{x}, \mathbf{y})$ = "the classifier switches from making a correct classification decision to an incorrect one on a point (x, y)"
- **Forgetting score of** $(\mathbf{x}, \mathbf{y})$ = the number of times during training when it was included in a minibatch and underwent a forgetting event (2 events simultaneously)

Note : From Toneva (2018), it turns out that examples with low forgetting score can be completely omitted during training without any noticeable effect on test accuracy.

Error L2-Norm score and forgetting score

Experiment setting

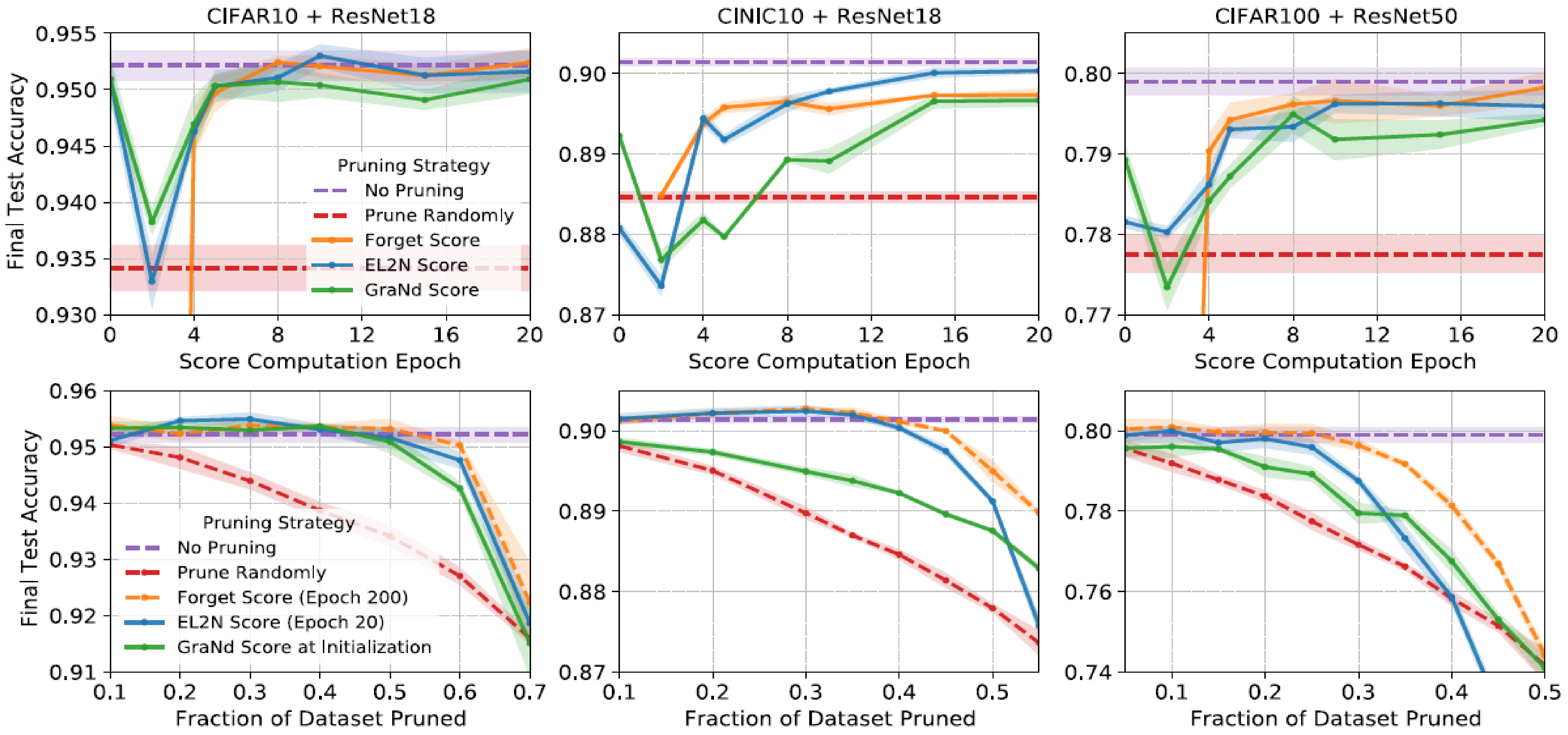
- 1st row exp : calculate data values using all train data (differ score computation epoch) -> Train with certain fraction of train data (pruned data) and test
- 2nd row exp : calculate data values using all train data (fix score computation epoch) -> Train with different fraction of train data (pruned data) and test

Procedure :

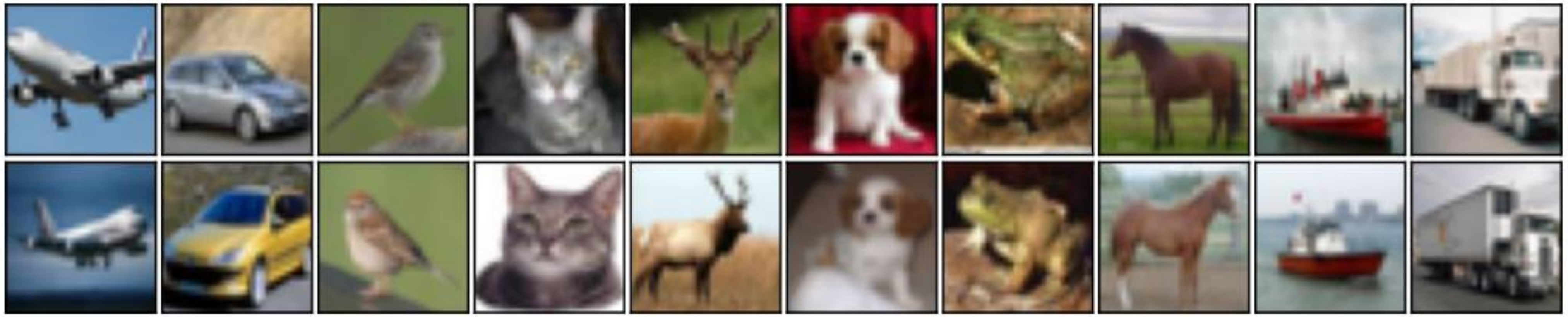
1. Train 10 independent models on each data
2. Calculate scores at both initialization or early training at given epoch and take expectation over 10 independent models
3. Train new randomly initialized model with pruned data (remove lowest score first)
4. Evaluate performance and repeat this procedure 4 times independently (to get percentile of result)

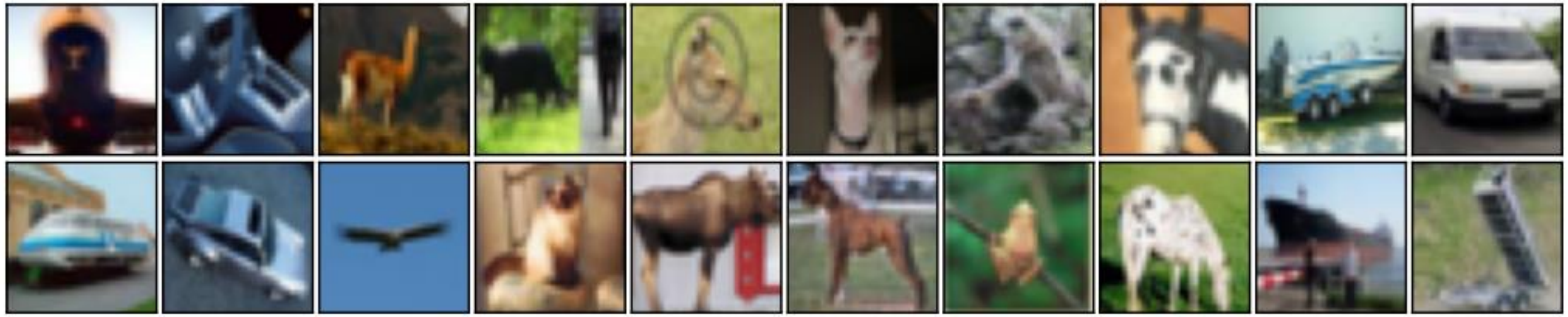
Error L2-Norm score and forgetting score

Data set + Network



Identifying noise examples



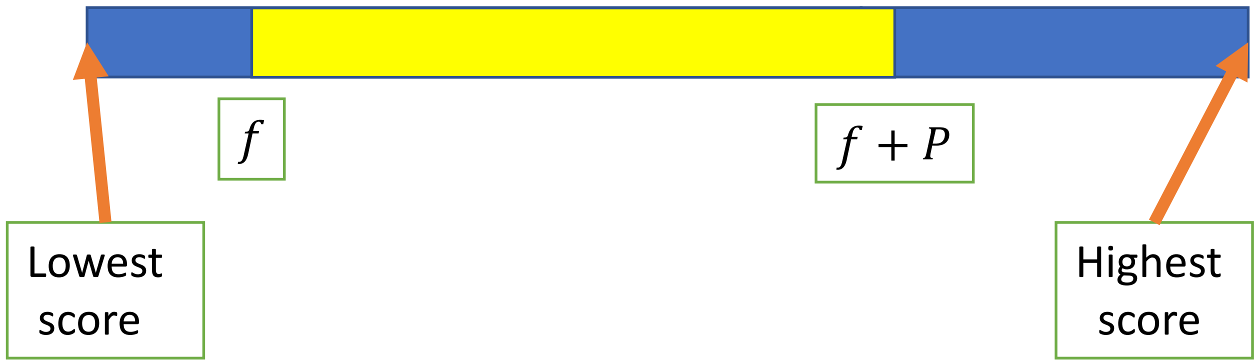


Examples with the 2nd lowest(above) - highest (below)/ 1st lowest - largest GraNd score for each class

(Left to right : airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck)

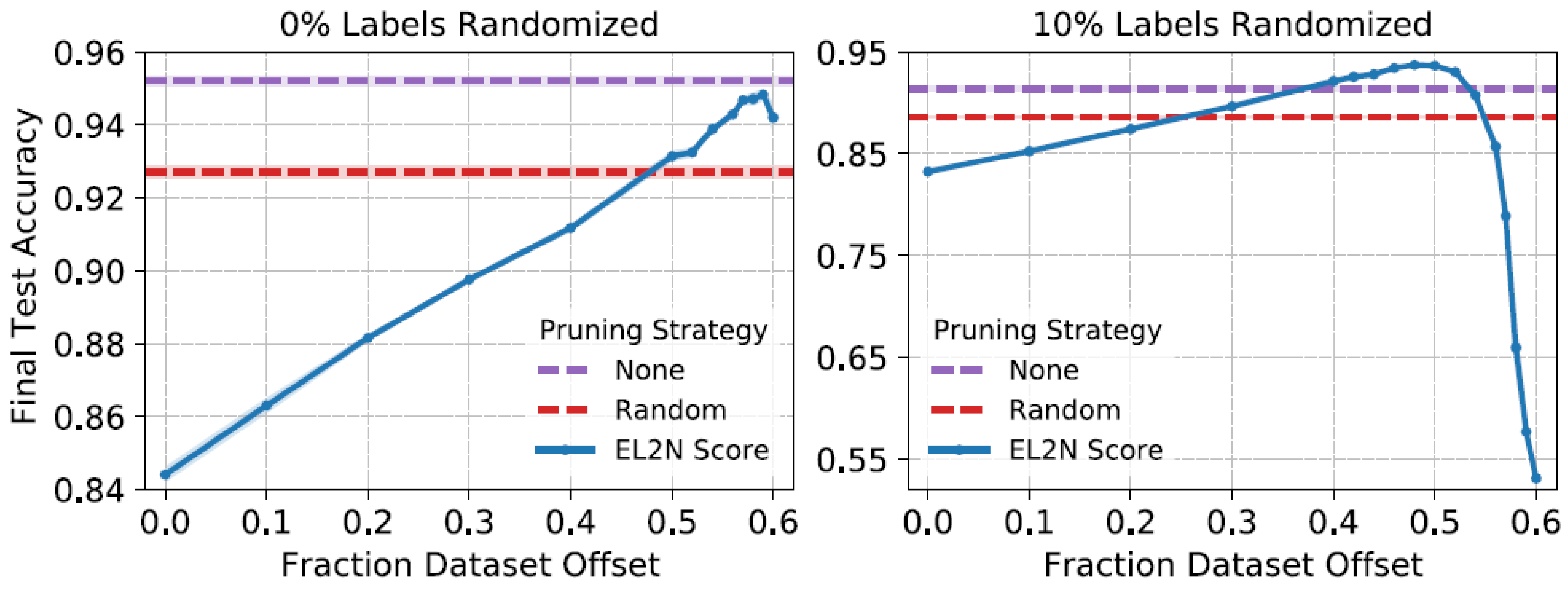
Does the highest-scoring examples are the most important ones for achieving an accurate classifier?

Identifying noise examples



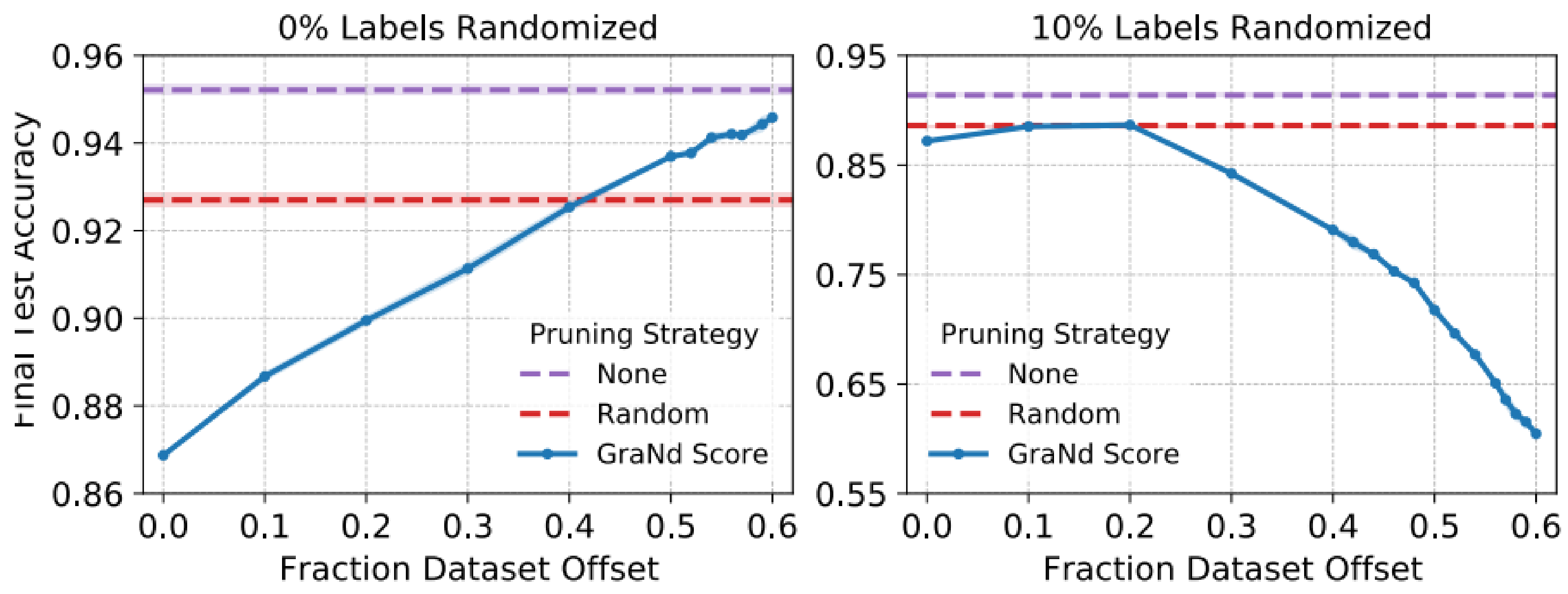
Train data with hyperparameter f, P
(f = offset, P = fixed portion (40%))

blue area data = pruned data
yellow area data = To-be-trained data

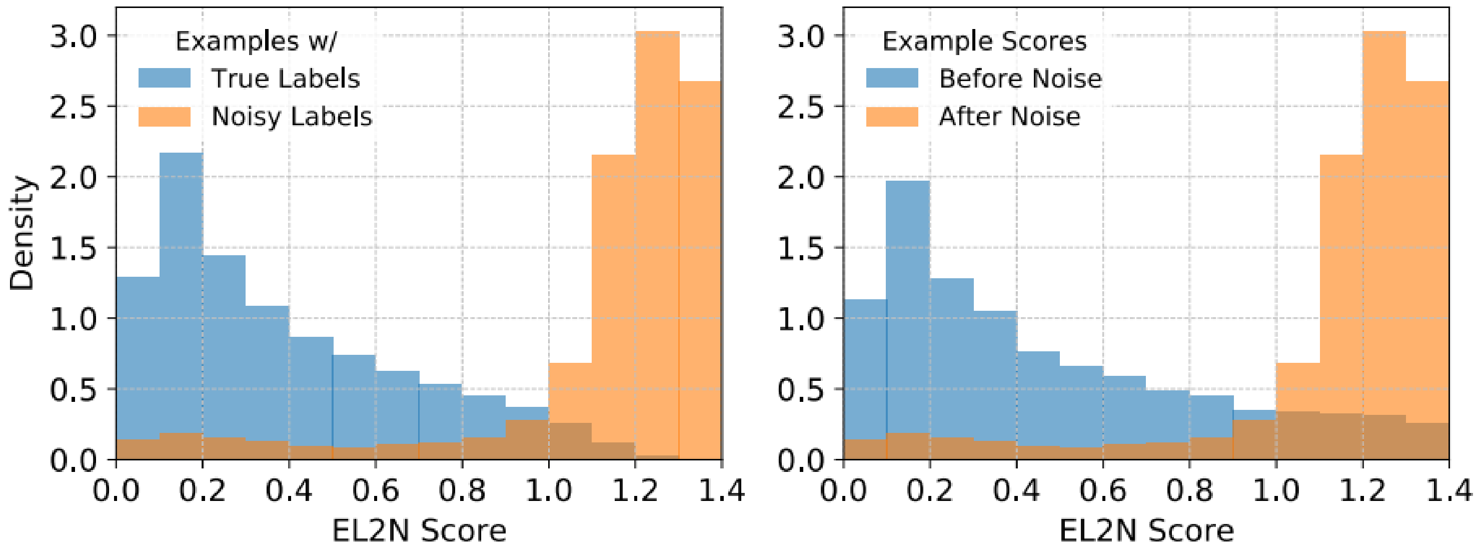


Model = ResNet18
Data = CIFAR-10
Score = EL2N
Score computation epoch = 10

Identifying noise examples



Model = ResNet18
Data = CIFAR-10
Score = GraNd
Score computation epoch = Initialization



distribution of EL2N scores :

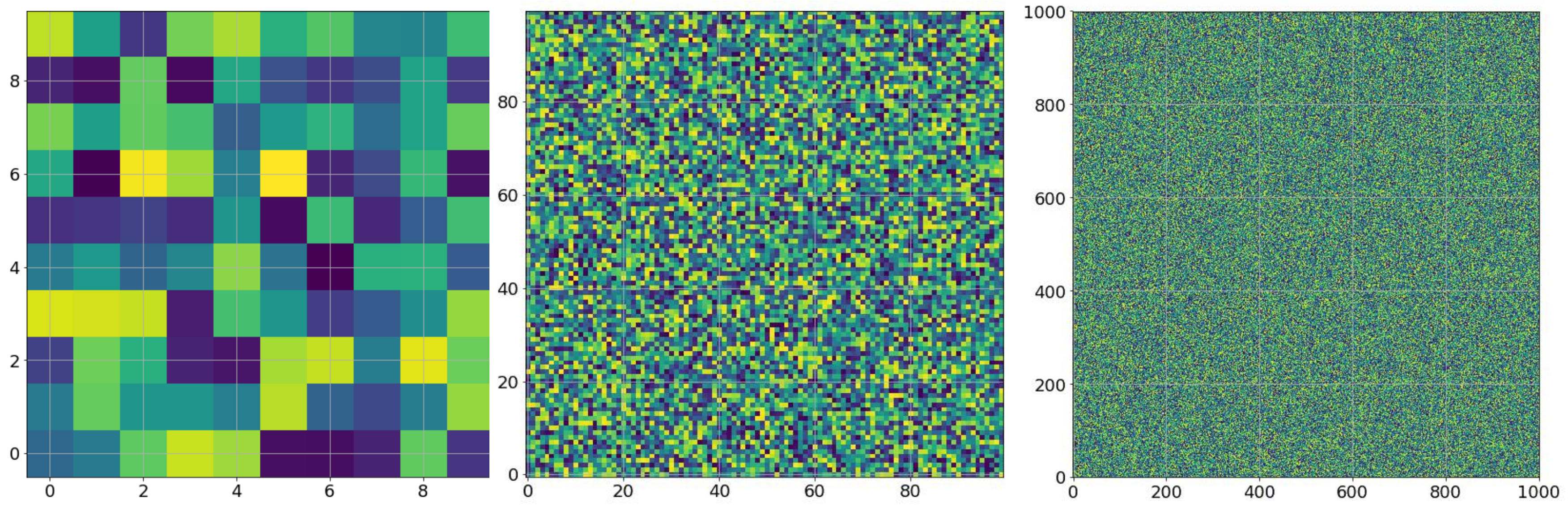
- True label – Noisy label (left)
- Before Noise – After noise (right)

Note : Impose Noise = Randomize

Evolution of the data-dependent NTK

Background : Neural Tangent Kernel (NTK)

- Assumption : one-layer NN with m neurons (m = width-size)
- **Intuition : As the width-size gets bigger, each weight changes during training gets smaller**
- Result : $f_w(x) \approx f_{w_0}(x) + \nabla_{\mathrm{w}} f(w_0, x)^T (w - w_0)$, and $K(x, x') = < \nabla_{\mathrm{w}} f(w_0, x), \nabla_{\mathrm{w}} f(w_0, x') >$ acts like a kernel in kernel method (called 'Neural Tangent Kernel') => NN implements kernel regression with the fixed NTK as kernel when $m \to \infty$.



Animation of weight change during training (m = 10, 100, 1000)

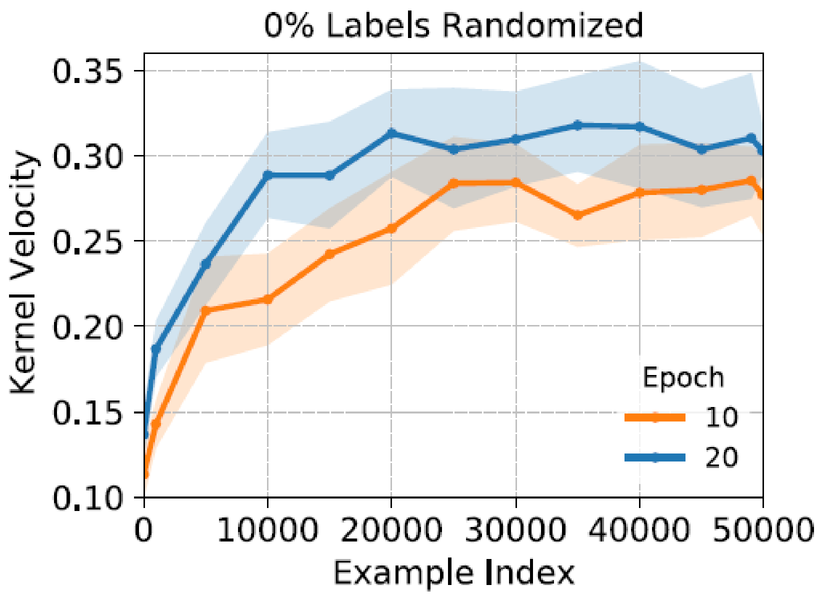
Evolution of the data-dependent NTK

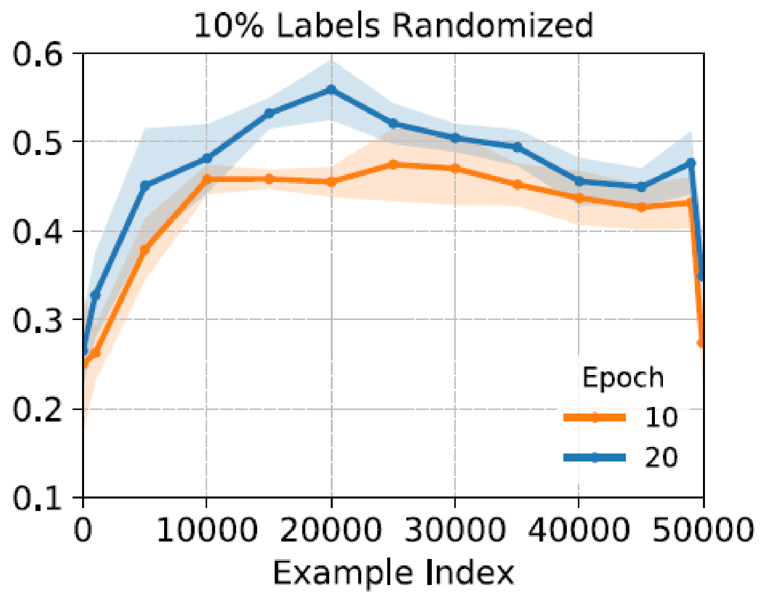
Background : Data-dependent Neural Tangent Kernel (NTK) (S.Fort, 2019)

- **Problem : finite NN have quite different weight dynamics early in training.**
- Suggestion : $f_{w_{t+1}}(x) \approx f_{w_t}(x) + \nabla_{\mathrm{w}} f(w_t, x)^T (w_{t+1} - w_t)$ and track the $\nabla_{\mathrm{w}} f(w_t, x)$
- Define data-dependent NTK : $K_t(x, x') = < \nabla_{\mathrm{w}} f(w_t, x), \nabla_{\mathrm{w}} f(w_t, x') >$
- Note : Kernel matrix K_t (= Gram matrix) is an $mC \times mC$ matrix where $[K_t]_{i,j} = K_t(x_i, x_j)$, where data points $\{x_1, \dots, x_m\}$ are given and C = the number of classes. (denote NTK matrix)
- How to describe the evolution of NTK? : By Kernel distance
- Kernel distance : $S(w_t, w_{t'}) = 1 - \frac{\ll K_t, K_{t'} \gg}{\|K_t\|_F \|K_{t'}\|_F}$ / Kernel velocity : $v(t) = \frac{S(w_t, w_{t+dt})}{dt}$ (Here, we set $dt = 1$)

  (Note : $\ll A, B \gg = Tr\left(A^T B\right)$ is Frobenius inner product)
- Result from suggested paper : test error barrier and kernel velocity are strongly correlated (positive)

Evolution of the data-dependent NTK





- Examples are sorted in ascending order by EL2N scores
- [Index : Index+100] data points are used to calculate data-dependent NTK matrix (f = index, $P = 100$)
- Score and velocities are calculated at given epoch (color)

Interpretation

- **Kernel velocity drops off sharply for examples with the very highest scores when label noise is introduced.**
- Suggested hypothesis : while the kernel velocity is higher for harder examples (the model is actively trying to fit), the kernel velocity drops off for the very highest scoring examples that might be too difficult to learn, or unrepresentative samples, and noisy samples

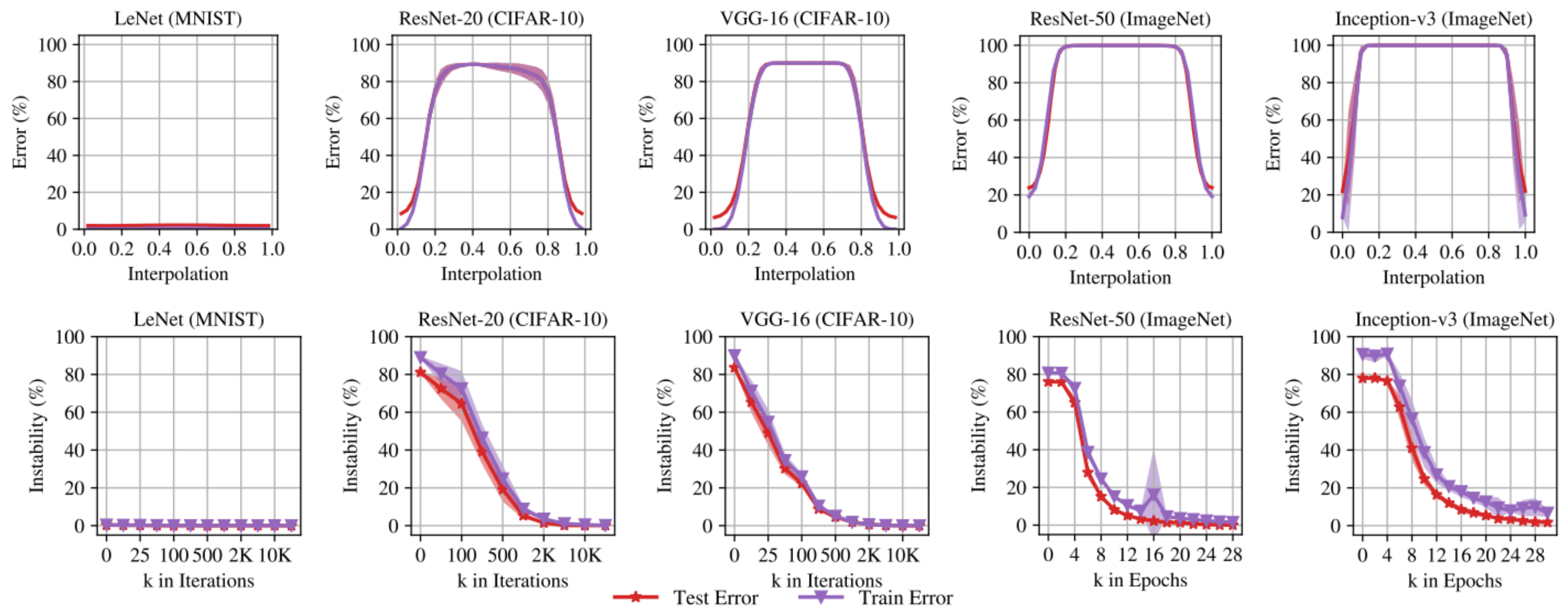
Connections to the Linear Mode Connectivity

Background : Linear mode connectivity (J.Frankle, 2020)

- **Problem :** When training a NN, SGD involves randomness (by different mini-batch shuffling or data augmentation) and alter trajectory of trained model = > Requires Instability analysis to check when the model gets stable to SGD noise during training
- **Suggested analysis method :**
  1. Train a model [parent network] up to step k (or epoch k)
  2. Copy (spawn) a same model W_2 [child network] with parent' weights and train parent and child until the desired step, but with different SGD noise.
  3. Denote the parent weight W_1, child weight W_2 and check test error of a network $\alpha W_1 + (1-\alpha)W_2$ for $\alpha \in [0,1]$ (=$\mathcal{E}(\alpha W_1 + (1-\alpha)W_2)$)
- **Measure for instability of Network : (called linear interpolation instability (or) error barrier height)**

$$\sup_{\alpha\in[0,1]} \mathcal{E}(\alpha W_1 + (1-\alpha)W_2) - \frac{1}{2}(\mathcal{E}(W_1) + \mathcal{E}(W_2))$$

Connections to the Linear Mode Connectivity



Background : Linear mode connectivity (J.Frankle, 2020)

- Two networks W_1 and W_2 are 'mode connected' if there exists a path between them along which the error barrier height ≈ 0. further, if the path is linear path, then they are 'linearly mode connected'.
- Result : As the spawn step k gets larger, the error barrier height $\rightarrow 0$

Connections to the Linear Mode Connectivity

Background : Linear mode connectivity (Paper Notation)

- Let $w_1, \dots w_T$ be the weight trajectory of a parent network.
- Fix a spawning time t^* and let $v_{t^*}, v_{t^*+1} \dots, v_T$ be an weight trajectory of a child network.

- Training error barrier between two weight w and w' (slight modification from original):

$$err(w, w'; S) = \sup_{\alpha \in [0,1]} \{\hat{R}_S(\alpha w + (1-\alpha) w') - (\alpha \hat{R}_S(w) + (1-\alpha)\hat{R}_S(w'))\}$$

where $\hat{R}_s(w)$ is training error of network with weight w

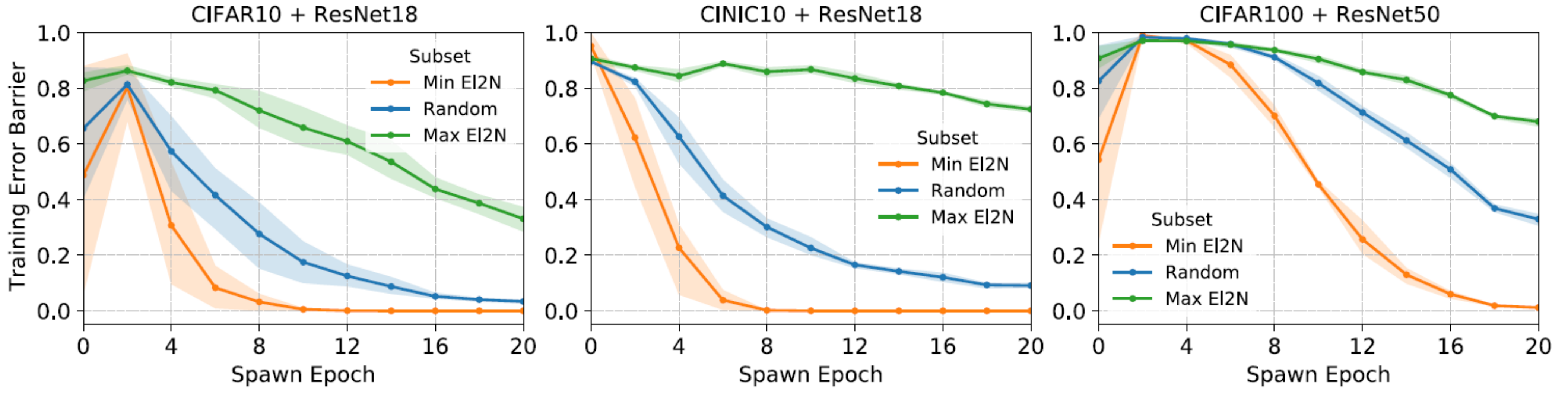
- Mean training error barrier spawning at t^*, at time t for $t^* \leq t \leq T$:

$$err_t^{t^*}(S) = \mathbb{E}_{w_{t^*+1:t}, v_{t^*+1:t}}[err(w_t, v_t; S)]$$

where expectation is taken over randomness in the trajectories of w and v after t^*

Connections to the Linear Mode Connectivity

Min EL2N = 1000 lower EL2N score data
Max EL2N = 1000 highest EL2N score data
Random = 1000 random data



Interpretation

- **Error barrier falls close to zero very rapidly for examples that have low EL2N scores, and stay high for high score examples.**
- Suggested hypothesis : Loss landscape derived from easy subsets of examples with low scores is quite flat. But, the loss landscape derived from harder subsets of examples with higher scores is rougher (by observing spawn epoch – Training error barrier graph)