Quantifying lottery tickets under label noise: accuracy, calibration, and complexity (Supplementary Material)

Viplove Arora[1] **Daniele Irto**[2] **Sebastian Goldt**[1] **Guido Sanguinetti**[1]

[1]Theoretical and Scientific Data Science, SISSA, Trieste, Italy
[2]Data Science and Scientific Computing, University of Trieste, Trieste, Italy

A MIXTURE CLASSIFICATION DATASETS

A common technique in the study of neural network theory is to operate in a controlled setting, where models and data are simplified to allow analytical computations for otherwise intractable objects. An example of such an approach is the teacher-student framework [Seung et al., 1992], where the dataset is created by a neural network built for that purpose (teacher) and another neural network is tasked with learning on that data (student). For our analysis, we create two types of balanced binary classification datasets using Gaussian mixtures with different characteristics and varying levels of difficulty. Using the mixture model allowed us to create datasets that are easy to understand and can be easily visualised in a two-dimensional space.

We consider two settings for mixture classification. The *linear* dataset consists of two clusters that can be separated using a linear function. The two clusters have different means $\mu_1 \neq \mu_2$ but same covariance $\Sigma_1 = \Sigma_2$. The *XOR* dataset was created such that the resulting clusters are placed like the graphic representation of the XOR logical function. We assume that the difficulty of the classification task would increase going from linear to XOR dataset. Data was sampled using a Gaussian mixture model conditioned on a predefined label y. We also define a coefficient of separation ν to modulate the distance between clusters.

To properly define the formulation of our mixture models, we use an approach similar to that used by Refinetti et al. [2021]. The data distribution for a single input sample $\mathbf{x} \in \mathbb{R}^D$ given class y sampled uniformly at random is:

$$p(\mathbf{x}, y) = p(y)\, p(\mathbf{x}|y), \quad p(\mathbf{x}|y) = \sum_{\alpha \in \mathcal{S}^T(y)} \mathcal{P}_\alpha \, \mathcal{N}(\mu_\alpha, \mathbf{\Sigma}_\alpha). \tag{S1}$$

where $p(\mathbf{x}|y)$ is the probability of sampling $\mathbf{x}$ conditioned on the class $y \in \{0, 1\}$ of the sample. Each $\mathbf{x}$ is sampled using a multivariate normal distribution. $\mathcal{S}^T(y)$ is the set of all possible indexes for class y and dataset type T. $\mathcal{P}_\alpha$ is the probability of the α-th D-dimensional multivariate normal distribution $\mathcal{N}(\mu_\alpha, \mathbf{\Sigma}_\alpha)$, which depends on the size of $\mathcal{S}^T(y)$. This formulation can easily be used to generate data with a generic number of clusters, classes, and dataset types. In particular, more classes and dataset types can be included by defining proper, additional sets of indexes $\mathcal{S}^T(y)$. For our experiments, we only considered two classes.

A.1 DATA GENERATION PROCESS

Labels: The first step in generating the data was to create a vector y of classes 0 or 1 of size equal to the desired number of training samples N. This was done by sampling N values from a uniform distribution $\mathcal{U}(0, 1)$ and reassigning them to values 0 or 1 depending on whether they were ≤ 0.5 or > 0.5, respectively. The resulting vector $\mathbf{y}$ is an array of values 0 or 1 in balanced proportions and it corresponds to the labels of the training samples in our dataset.

Linear dataset: For each observation $\mathbf{x_i}$, $i = 1, \ldots, N$, of the linear dataset, the set of indexes $\mathcal{S}^L(y)$ has only one element for each class. This means that, for each class, the input points can be sampled by one multivariate normal:

$$\mathcal{S}^L(y=0) = \{\alpha_0\} \;\rightarrow\; \mu_{\alpha_0} = 0 \cdot \mathbb{1}^D \tag{S2a}$$

$$\mathcal{S}^L(y=1) = \{\alpha_1\} \;\rightarrow\; \mu_{\alpha_1} = \nu \cdot \mathbb{1}^D \tag{S2b}$$

$$\mathbf{\Sigma}_{\alpha_0} = \mathbf{\Sigma}_{\alpha_1} = I_D. \tag{S2c}$$

$\mathbb{1}^D$ is a D-dimensional vector of all ones that can be multiplied by a scalar, meaning that all its elements get multiplied by that scalar. I_D is the $D \times D$ identity matrix, and its elements can be multiplied by a scalar number as well[1]. The distance between the two clusters, which makes the two classes more or less discernible, can be changed by simply increasing or decreasing the value of the ν coefficient. Performing PCA on the linear dataset and plotting the first two principal components yields the clusters shown in fig. 2a. In those plots, it is possible to observe a clear distinction between the clusters belonging to the two classes. We can also see the change in distance between the clusters as ν is changed.

XOR dataset: For the second dataset, our goal was to make the data appear as four separate clusters placed like the visual representation of the XOR logical operator. In this case, the sets of indexes corresponding to class $y = 0$ and $y = 1$ that have two elements each. Thus, the data points of each class can be sampled from two different distributions with equal probabilities. For class $y = 0$, the samples are generated like the linear dataset described in eq. (S2). For the other class, its main feature is that the mean vectors of the multivariate normal distributions consist of two $\frac{D}{2}$-dimensional halves with different values:

$$\mathcal{S}^X(y=0) = \{\alpha_0^A, \alpha_0^B\} \;\rightarrow \begin{cases} \mu_{\alpha_0^A} = 0 \cdot \mathbb{1}^D \\ \mu_{\alpha_0^B} = \nu \cdot \mathbb{1}^D \end{cases} \tag{S3a}$$

$$\mathcal{S}^X(y=1) = \{\alpha_1^A, \alpha_1^B\} \;\rightarrow \begin{cases} \mu_{\alpha_1^A} = [0 \cdot \mathbb{1}^{\frac{D}{2}}\,,\, \nu \cdot \mathbb{1}^{\frac{D}{2}}] \\ \mu_{\alpha_0^B} = [\nu \cdot \mathbb{1}^{\frac{D}{2}}\,,\, 0 \cdot \mathbb{1}^{\frac{D}{2}}] \end{cases} \tag{S3b}$$

$$\mathbf{\Sigma}_{\alpha_0^A} = \mathbf{\Sigma}_{\alpha_0^B} = \mathbf{\Sigma}_{\alpha_1^A} = \mathbf{\Sigma}_{\alpha_1^B} = I_D. \tag{S3c}$$

The PC plots of this dataset are shown in fig. 2b, where we can see the positioning of the four clusters in a cross-like layout.

B PRUNING AND REWINDING

In our experiments, training is always followed by a series of pruning iterations that were performed according to the IMP [Han et al., 2015, Frankle and Carbin, 2019] technique, which is described below:

1. Initialise the model with weight $\mathcal{W}_0$, obtaining a model function $f(x\,;\,\mathcal{W}_0)$.
2. Train the model for the desired number of epochs j, reaching a set of weights $\mathcal{W}_j$.
3. Sort all the weights in $\mathcal{W}_j$ by their absolute value and select the lowest p%, where p is a number between 0 and 100.
4. Prune the selected weights by applying a mask to the original model, obtaining $f(x\,;\,\text{mask} \odot \mathcal{W}_j)$.
5. Reset the remaining parameters to their first initialisation values, obtaining $f(x\,;\,\text{mask} \odot \mathcal{W}_0)$.
6. Iterate the process n times, pruning $p\%$ of the remaining connections at each iteration.

In certain cases, the larger models need to be pruned using the lottery ticket rewinding technique [Frankle et al., 2019]. Rewinding simply consists of modifying only one step of the procedure described above:

5 Reset the remaining parameters to the values at iteration $k \ll j$ of the training loop, obtaining $f(x\,;\,\text{mask} \odot \mathcal{W}_k)$

where k is a hyperparameter representing the number of the rewinding iterations. This technique is also conveniently included in the `OpenLTH`[2] library.

[1]This notation is used consistently in this section, to indicate the means and covariances of the normal distributions.

[2]`https://github.com/facebookresearch/open_lth`

C MODEL HYPERPARAMETERS

C.1 MIXTURE CLASSIFICATION

We used two-layer fully connected networks without bias for our experiments on the two mixture classification datasets. Models with $P \in \{1, 3, 5, 8, 10, 15, 20, 25, 50, 100, 200, 500, 1000, 10000\}$ neurons in the hidden layer were used to produce the double descent curve. These models were subsequently pruned using IMP to produce the sparse double descent curves. Network weights are initialised using the Kaiming uniform distribution. The activation function chosen for this model is the Rectified Linear Unit (ReLU). Stochastic gradient descent with a learning rate of 0.1 was used as the optimiser. All models were trained for 1000 epochs using a batch size of 1024. All models were sufficiently pruned to observe the sparse double descent curves. All experiments were replicated five times.

Linear datasets: We varied the distance between cluster means $\nu \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ to see how it impacts the test error and the effective number of parameters. 5% random label noise in the training set was used to observe the two double descent curves.

XOR datasets: We varied the distance between cluster means $\nu \in \{0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$ to see how it impacts the test error and the effective number of parameters. Higher values of ν were needed to ensure sufficient distance between the clusters. Higher label noise of 25% was needed to consistently observe the two double descent curves in the XOR datasets.

C.2 MNIST

The three-layer fully connected architecture used for MNIST is an extension of the standard model used for MNIST in the pruning literature. We varied the number of neurons $P \in \{3, 5, 10, 25, 50, 100, 300, 500, 1000, 5000, 10000\}$ in the first hidden layer. The size of the second hidden layer was kept fixed at 100. For the two-layer FCN, $P \in \{5, 10, 50, 100, 300, 500, 1000, 2000, 5000, 10000\}$ neurons were used in the hidden layer. For the fully connected networks, lottery ticket rewinding was used for networks with $P \geq 1000$. For ResNet-6, we varied the width $W \in \{1, 2, 5, 8, 11, 15, 20, 40, 80, 120\}$ of the convolutional filters to obtain networks of different sizes. Further details can be found in table S1.

C.3 FASHION-MNIST

We also performed a small set of experiments on the Fashion-MNIST dataset. We only performed a limited number of experiments focused on finding the effective number of parameters using overparameterised models. Since reproducing the double descent curve was not the target, we only used three-layer FCNs with $P \in \{500, 1000\}$ neurons in the first hidden layer. The size of the second hidden layer was kept fixed at 100. Note that more epochs (320) are needed to train a three-layer FCN on Fashion-MNIST.

C.4 CIFAR-10

We primarily considered ResNet-18 for CIFAR-10, where the width $W \in \{2, 5, 8, 11, 15, 20, 40, 60, 80, 100, 120, 150\}$ of the convolutional filters was varied to obtain networks of different sizes. Based on previous observations Frankle et al. [2019], He et al. [2022], we used 10 epochs of rewinding to consistently find lottery tickets in ResNet-18. To compare the effective number of parameters across different CNN architectures, we performed our analysis on DenseNet-121 and VGG-16. Further details can be found in table S1.

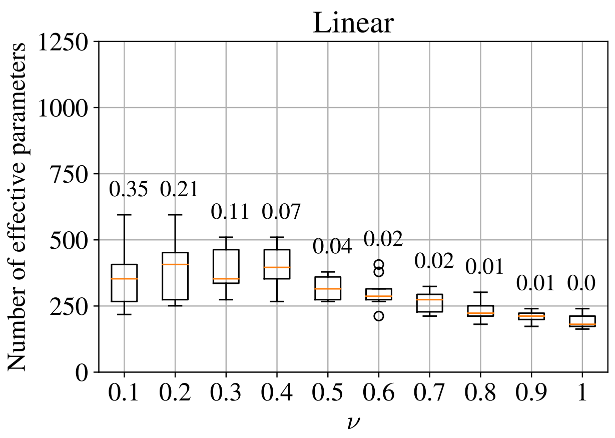
D ADDITIONAL RESULTS

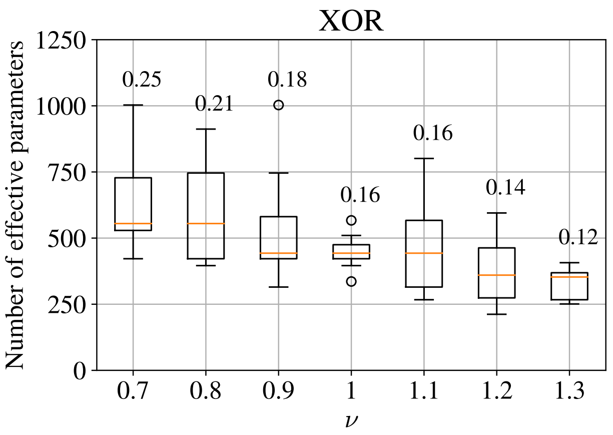
Figure S1 shows how the effective number of parameters and test error of the best pruned models vary with ν for the linear and XOR datasets. As expected, the test error decreases with the distance between cluster means ν. For networks trained on the linear dataset, we find that the effective number of parameters, i.e. the number of parameters in the best pruned models, ranges from ~ 200 to ~ 500 for different starting models and for different values of ν (see appendix D for the full

Table S1: Neural network architectures used on real data. LR refers to the learning rate.

Network	Dataset	Epochs	Batch size	Optimiser	Momentum	LR	Rewind Iter
Two-layer FCN	MNIST	120	128	SGD	-	0.1	-
Three-layer FCN	MNIST	120	128	SGD	-	0.1	-
ResNet-6	MNIST	120	128	SGD	0.9	0.1	-
Three-layer FCN	Fashion MNIST	320	128	SGD	-	0.1	-
ResNet-18	CIFAR-10	160	128	SGD	0.9	0.1	10 epochs
VGG-16	CIFAR-10	160	128	SGD	0.9	0.1	10 epochs
DenseNet-121	CIFAR-10	160	128	SGD	0.9	0.1	10 epochs

distribution). We find the same behaviour for the XOR dataset, fig. 2b, but with a higher number of parameters (between 250 and 1000) than for the linear dataset with the same ν.





(a) XOR datasets.

Figure S1: Distribution of the effective number of parameters of the best pruned models (y-axis) as the distance between the clusters ν (x-axis) is varied for the linear and xor datasets. Only pruned models originating from overparameterised full models are considered. The numbers above the boxes report the test error of the model with median effective number of parameters for each ν.

Double descent and sparse double descent curves obtained for MNIST on two-layer FCN and ResNet-6 can be seen in fig. S2. Table S2 shows the number of parameters and test errors for the full/unpruned models and the corresponding best pruned models. Sparse double descent curves for two different models on Fashion-MNIST in fig. S3 show that, compared to MNIST, the test error for the best pruned models is higher while the effective number of parameters is approximately 10 000.

Finally, the calibration curves when label noise is added to the test set for MNIST and CIFAR-10 can be seen in fig. S4. The class-averaged calibration curves show that the pruned models are well-calibrated to noisy data while the full models are overconfident.

REFERENCES

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

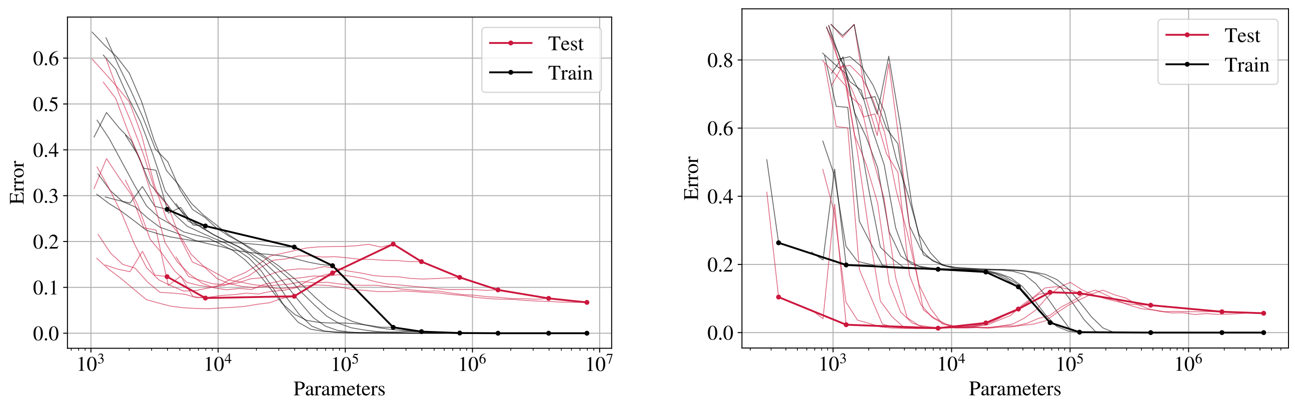


Figure S2: Pruning models along the double descent curve (dark red) show that sparse double descent curves (light red) from different models coincide at the minima. Results are shown for two-layer FCNs (left) and ResNet-6 (right) on MNIST with 20% label noise averaged over three replicates.

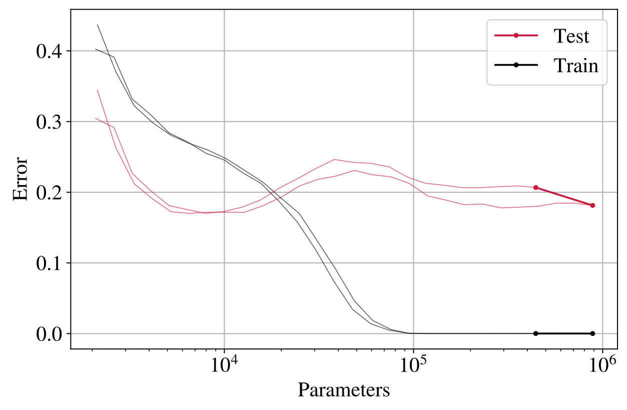


Figure S3: Estimating the effective number of parameters for Fashion-MNIST: Results are shown for three-layer FCNs on Fashion-MNIST with 20% label noise averaged over three replicates. Compared to MNIST, the test error for the best pruned models is higher while the effective number of parameters is approximately $10\,000$.

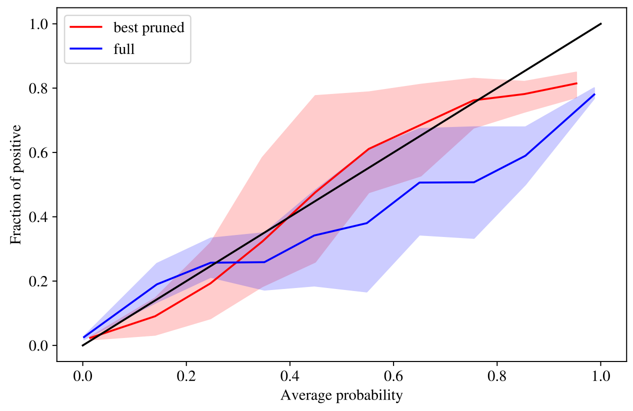
Table S2: Number of parameters and test error for unpruned and best pruned models for two architectures trained on MNIST: two-layer FCNs, and ResNet-6. Average values over 3 replicates are reported. We observe that a $200\times$ increase for the full models results in only a $\sim 3.5\times$ increase in the number of parameters for the best pruned models. Notice also that the error achieved by pruned models appears insensitive to the error rate of the original full model, i.e. even models with poor generalisation can be rescued by pruning.

2 layer FC				ResNet-6			
Parameters		Test error		Parameters		Test error	
Full	Pruned	Full	Pruned	Full	Pruned	Full	Pruned
39 700	8320	0.081	0.053	19 464	5098	0.028	0.012
79 400	6814	0.131	0.077	36 597	7669	0.069	0.012
238 200	5358	0.194	0.092	67 785	7272	0.118	0.014
794 000	11 436	0.122	0.106	120 180	8252	0.116	0.013
1 588 000	11 710	0.094	0.095	478 760	13 470	0.080	0.013
3 970 000	23 428	0.076	0.103	1 911 120	17 620	0.061	0.014
7 940 000	29 990	0.067	0.097	4 297 080	20 286	0.057	0.013

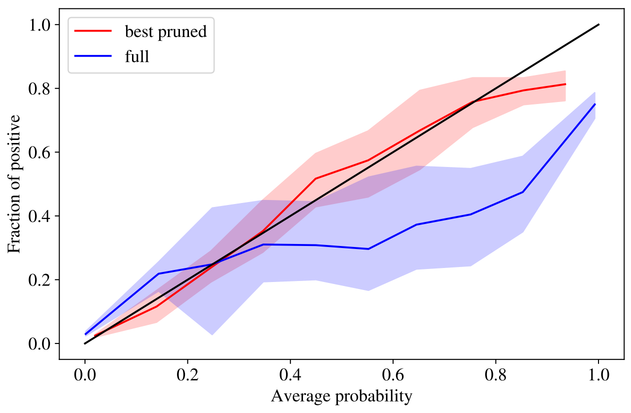
Zheng He, Zeke Xie, Quanzhi Zhu, and Zengchang Qin. Sparse double descent: Where network pruning aggravates overfitting. In *International Conference on Machine Learning*, pages 8635–8659. PMLR, 2022.

Table S3: Effective number of parameters and test error of best pruned models on subsets of CIFAR-10 dataset.

Classes	Params	Error
5	16 312	0.102
6	20 392	0.118
7	31 869	0.122
8	31 871	0.125
9	31 872	0.126
10	39 843	0.124



(a) MNIST with two-layer FCN. ECE for pruned and full models are 0.075 and 0.207, respectively.



(b) CIFAR-10 with ResNet-18. ECE for pruned and full models are 0.053 and 0.254, respectively.

Figure S4: Class-averaged calibration curves for the best pruned and full models on (a) MNIST and (b) CIFAR10 datasets with noise added to the labels in the test set show that the pruned models are well-calibrated to noisy data while the full models are overconfident. The highlighted areas signify deviation between classes.

Maria Refinetti, Sebastian Goldt, Florent Krzakala, and Lenka Zdeborová. Classifying high-dimensional gaussian mixtures: Where kernel methods fail and neural networks succeed. In *International Conference on Machine Learning*, pages 8936–8947. PMLR, 2021.

Hyunjune Sebastian Seung, Haim Sompolinsky, and Naftali Tishby. Statistical mechanics of learning from examples. *Physical review A*, 45(8):6056, 1992.