

## REFERENCES

- Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.
- Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.
- Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.
- Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.
- Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *International conference on computer vision*, 2019.
- Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
- Don Dennis, Chirag Pabbaraju, Harsha Vardhan Simhadri, and Prateek Jain. Multiple instance learning for efficient sequential data classification on resource-constrained devices. *Advances in Neural Information Processing Systems*, 31, 2018.
- Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- Yoav Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory, COLT ’90*, pp. 202–216, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc. ISBN 1558601465.
- Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. ISSN 0022-0000. doi: <https://doi.org/10.1006/jcss.1997.1504>. URL <https://www.sciencedirect.com/science/article/pii/S002200009791504X>.
- Mengya Gao, Yujun Wang, and Liang Wan. Residual error based knowledge distillation. *Neurocomputing*, 433:154–161, 2021.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.
- Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 2021.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.
- Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017a.
- Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017b.

- Mikhail Khodak, Renbo Tu, Tian Li, Liam Li, Maria-Florina Balcan, Virginia Smith, and Ameet Talwalkar. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing, 2021.
- Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL <http://www.cs.toronto.edu/~kriz/cifar.html>.
- Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.
- Gábor Lugosi and Nicoló Cesa-Bianchi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
- Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. *Advances in neural information processing systems*, 12, 1999.
- Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *AAAI Conference on Artificial Intelligence*, 2020.
- Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.
- Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.
- Adria Ruiz and Jakob Verbeek. Anytime inference with distilled hierarchical neural ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.
- Robert E Schapire. The strength of weak learnability. *Machine learning*, 1990.
- Robert E Schapire and Yoav Freund. Boosting: Foundations and algorithms. *Kybernetes*, 2013.
- Arun Suggala, Bingbin Liu, and Pradeep Ravikumar. Generalized boosting. *Advances in neural information processing systems*, 33:8787–8797, 2020.
- Luca Trevisan, Madhur Tulsiani, and Salil Vadhan. Regularity, boosting, and efficiently simulating every high-entropy distribution. In *2009 24th Annual IEEE Conference on Computational Complexity*, 2009.
- John von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*, 1944.
- Pete Warden. *Speech commands: A dataset for limited-vocabulary speech recognition*, 2018.
- Li Yang and Deliang Fan. Dynamic neural network to enable run-time trade-off between accuracy and latency. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. Association for Computing Machinery, 2021.

## A TWO-PLAYER MINIMAX GAMES

In this section, we will look at the setting of two minimax games more closely.

Consider a class of hypotheses  $\mathcal{F}$  and class of probability distributions  $\mathcal{P}$ . In addition, consider a loss function  $\mathcal{L} : \mathcal{F} \times \mathcal{X} \rightarrow \mathbb{R}$  that is convex in its first argument. We consider two players whose pure strategy sets are  $\mathcal{F}$  and  $\mathcal{P}$  respectively. The loss and reward of the players is given by  $F(f, \mu) = \mathbb{E}_{x \sim p}[L(f, x)]$  and the minimax value of the game is

$$\max_{p \in \mathcal{P}} \min_{f \in \mathcal{F}} F(f, \mu). \quad (11)$$

Note that this game is convex in the hypothesis player and concave in the distribution player. The objective of the game for the hypothesis player is trying to find a hypothesis that has low loss on the worst case distribution from class  $\mathcal{P}$ . Conversely, the distribution player is trying to construct a distribution that is as hard as possible for the hypothesis player to learn.

Also, note that under reasonable conditions on  $\mathcal{F}$  and  $\mathcal{P}$ , we have the minimax theorem holds, see (Schapire & Freund, 2013, Chapter 6)

$$\max_{p \in \mathcal{P}} \min_{f \in \mathcal{F}} \mathbb{E}_{x \sim p}[L(f, x)] = \min_{f \in \Delta(\mathcal{F})} \max_{p \in \mathcal{P}} \mathbb{E}_{x \sim p}[L(f, x)]. \quad (12)$$

Here,  $\Delta(\mathcal{F})$  is the set distributions over functions  $\mathcal{F}$ . From this, we can see that as long as we are allowed to aggregate functions from the base class, we have can do as well as we could if we had access to the distribution

The interesting algorithmic question would be to find a distribution over hypothesis that achieves the minimum above. Note that the loss function is stochastic and thus, we need to formalize the access the player has to the loss function. In applications of interest for us, we focus on settings where We will formulate this in the following stochastic way.

**Definition 2.** An algorithm ORACLE is said to be a  $(\beta, \delta)$  weak-gradient if

$$\langle \text{ORACLE}(f, p), \nabla_f F(f, p) \rangle \geq \beta \quad (13)$$

with probability  $1 - \delta$ .

Here  $\nabla_f$  denotes the functional gradient of  $F$ . This notion is similar to the weak learning assumptions usually used in the boosting literature. Given such an oracle one can ask for methods similar to first order methods for convex optimization, such as gradient descent, to solve the minimax problem. These algorithms iteratively maintain candidate solutions for the both the players and update each of these using feedback from the state of the other player. In our particular setting, the hypothesis player updates using the vector  $h$  in eq. (2).

**Motivating Example** Let  $\mathcal{F}$  be a class of hypotheses, let  $\mathcal{P}$  is the set of all distributions over a finite sample set  $\{x_1, \dots, x_n\}$  and let  $L$  be the 0-1 loss. Note that in this setting, the oracle from definition 2 is analogous to weak learning. In this setting, from the minimax theorem and the existence of a weak learner, we get that there is a single mixture of hypothesis of  $\sum_i \alpha_i f_i$  such that loss under every distribution in  $\mathcal{P}$  which corresponds to zero training error. Thus we can think of boosting algorithms as approximating this minimax equilibrium algorithmically. Similarly, the weak learning condition in Suggala et al. (2020) is similar in spirit to the condition above.

With the condition from Definition 2, one can consider many frameworks for solving minimax games. One high general technique is to consider two no-regret algorithms for online convex optimization to play against each other. Let us briefly look at the definition of regret in this setting.

**Definition 3 (No-Regret).** Let  $K, A$  be convex sets. At each time, a player observes a point  $x_t \in K$  and chooses an action  $a_t \in A$ . The regret of the algorithm is defined as

$$R_T = \max_{a \in A} \sum_{t=1}^T \langle a, x_t \rangle - \sum_{t=1}^T \langle a_t, x_t \rangle. \quad (14)$$

Online learning is a well-studied area of machine learning with a rich set of connections to various areas in mathematics and computer science. In particular, there are frameworks in order to construct

algorithms such as follow-the-perturbed leader, follow-the-regularized leader and mirror descent. In particular, our algorithm can be seen as a version of mirror descent with the entropy regularizer and theorem 1 as a version of the regret guarantee for the algorithm. In addition to the ones mentioned above, There are several other frameworks considered to solve minimax games such as variational inequalities, extragradient methods, optimistic methods and so on. We believe this framework is a useful one to consider for many learning tasks, especially in settings where we have function approximation.

## B PROOFS

Here, we provide a proof of Theorem 1, which is restated below:

**Theorem.** Suppose the class  $\mathcal{F}$  satisfies that for all  $f \in \mathcal{F}$ ,  $\|f - g\|_\infty \leq G_\infty$ . Let  $F = \{f_t\}$  be the ensemble after  $T$  rounds of Algorithm 1, with the final output  $F_t = \frac{1}{T} \sum_{t=1}^T f_t$ . Then for  $T \geq \ln 2N$  and

$$\eta = \frac{1}{G_\infty} \sqrt{\frac{\ln 2N}{T}}$$

we have for all  $j$

$$\|F_{t,j} - g_j\|_\infty \leq G_\infty \sqrt{\frac{\ln 2N}{T}} - \frac{1}{T} \sum_{t=1}^T \gamma_t(j)$$

where  $F_{t,j}$  and  $g_j$  are the  $j^{\text{th}}$  coordinates of the functions  $F_t$  and  $g$  respectively.

*Proof.* For simplicity, we assume that  $f_t$  and  $g$  are scalar valued functions, since the proof goes through coordinate-wise. At each time, define the edge of the weak learning algorithm to be

$$\gamma_t = \sum_i K_t^+(i)(f_t(x_i) - g(x_i)) + \sum_i K_t^-(i)(g(x_i) - f_t(x_i))$$

Let  $Z_t$  denote the normalizing constant at time  $t$ , that is,

$$Z_t = \sum_i K_t^+(i) \exp(-\eta(f_t(x_i) - g(x_i))) + K_t^-(i) \exp(\eta(f_t(x_i) - g(x_i)))$$

From the update rule, we have

$$\begin{aligned} K_{T+1}^+(i) &= \frac{K_T^+(i) e^{\eta(f_T(x_i) - g(x_i))}}{Z_T} \\ &= \frac{K_1^+(i) \exp\left(-\eta \sum_{t=1}^T (f_t(x_i) - g(x_i))\right)}{\prod_{t=1}^T Z_t} \\ &= \frac{K_1^+(i) \exp(-\eta T(F_T(x_i) - g(x_i)))}{\prod_{t=1}^T Z_t} \end{aligned}$$

and similarly

$$K_{T+1}^-(i) = \frac{K_1^-(i) \exp(\eta T(F_T(x_i) - g(x_i)))}{\prod_{t=1}^T Z_t}$$

First, we bound  $\ln(Z_t)$ :

$$\begin{aligned}
\ln(Z_t) &= \ln \left( \sum_i K_t^+(i) \exp(-\eta(f_t(x_i) - g(x_i))) + \sum_i K_t^-(i) \exp(\eta(f_t(x_i) - g(x_i))) \right) \\
&\leq \ln \left( \sum_i K_t^+(i) (1 - \eta(f_t(x_i) - g(x_i)) + \eta^2(f_t(x_i) - g(x_i))^2) \right. \\
&\quad \left. + \sum_i K_t^-(i) (1 + \eta(f_t(x_i) - g(x_i)) + \eta^2(f_t(x_i) - g(x_i))^2) \right) \\
&\leq \ln \left( 1 - \eta \sum_i K_t^+(i)(f_t(x_i) - g(x_i)) + \eta \sum_i K_t^-(i)(f_t(x_i) - g(x_i)) + \eta^2 G_\infty^2 \right) \\
&\leq -\eta \gamma_t + \eta^2 G_\infty^2
\end{aligned}$$

where the second step follows from the identity  $\exp(x) \leq 1 + x + x^2$  for  $x \leq 1$ , provided that  $\eta \leq \frac{1}{G_\infty}$ . This gives us a bound on regression error after  $T$  rounds:

$$\begin{aligned}
-\eta T (F_T(x_i) - g(x_i)) &= \ln(K_{T+1}^+(i)) - \ln(K_1^+(i)) + \sum_{t=1}^T \ln(Z_t) \\
&\leq \ln \left( \frac{K_{T+1}^+(i)}{K_1^+(i)} \right) + \sum_{t=1}^T -\eta \gamma_t + \eta^2 G_\infty^2 \\
&= \ln \left( \frac{K_{T+1}^+(i)}{K_1^+(i)} \right) + \eta^2 T G_\infty^2 - \eta \sum_{t=1}^T \gamma_t \\
&\leq \ln 2N + \eta^2 T G_\infty^2 - \eta \sum_{t=1}^T \gamma_t
\end{aligned}$$

Where the last bound follows since  $K_1^+ = \frac{1}{2N}$  and  $K_{T+1}^+ \leq 1$ . Similarly, we have the bound

$$\eta T (F_T(x_i) - g(x_i)) \leq \ln 2N + \eta^2 T G_\infty^2 - \eta \sum_{t=1}^T \gamma_t$$

Combining the two equations we get that

$$\sup_i |F_T(x_i) - g(x_i)| = \|F_T - g\|_\infty \leq \frac{\ln 2N}{\eta T} + \eta G_\infty^2 - \frac{1}{T} \sum_{t=1}^T \gamma_t$$

If we choose  $\eta = \frac{1}{G_\infty} \sqrt{\frac{\ln 2N}{T}}$  to minimize this expression, then we get the following bound on regression error:

$$\|F_T - g\|_\infty \leq -\frac{1}{T} \sum_{t=1}^T \gamma_t + G_\infty \sqrt{\frac{\ln 2N}{T}}$$

which is exactly Equation (10). Note that the value of  $\eta$  only satisfies the condition  $\eta \leq \frac{1}{G_\infty}$  when  $T \geq \ln 2N$ , which is the time horizon after which the bound holds. This finishes the proof of Theorem 1  $\square$

Now, we provide a proof of Theorem 2 which follows from the VC dimension bound and Theorem 1. Before we begin, we setup some notation. Given a function  $f$ , distribution  $\mathcal{D}$  over space  $\mathcal{X} \times \mathcal{Y}$  where  $\mathcal{X}$  is the input space and  $\mathcal{Y}$  is the label space, and data  $D$  consisting of  $N$  iid samples  $(x, y) \sim \mathcal{D}$ , we define

$$\widehat{\text{err}}(f) = \Pr_{(x,y) \sim D} [\text{sign}(F_T(x) \neq y)] \quad \text{err}(f) = \Pr_{(x,y) \sim \mathcal{D}} [\text{sign}(F_T(x) \neq y)]$$

**Theorem (Excess Risk).** Suppose data  $D$  contains of  $N$  iid samples from distribution  $\mathcal{D}$ . Suppose that the function  $g$  has large margin on data  $D$ , that is

$$\Pr_{x \sim D} [|g(x)| < \epsilon] < \phi$$

Further, suppose that the class  $\mathcal{C}_T$  has VC dimension  $d$ , then for

$$T \geq \frac{4G_\infty^2 \ln 2N}{\epsilon^2}$$

, with probability  $1 - \delta$  over the draws of data  $D$ , the generalization error of the ensemble  $F_T$  obtained after  $T$  round of Algorithm 1 is bounded by

$$\text{err}(F_T) \leq \widehat{\text{err}}(g) + O\left(\sqrt{\frac{d \ln(N/d) + \ln(1/\delta)}{N}}\right) + \phi$$

*Proof.* Recall the following probability bound (Schapire & Freund, 2013, theorem 2.5) which follows Sauer’s Lemma:

$$\Pr [\exists f \in \mathcal{C}_T : \text{err}(f) \geq \widehat{\text{err}}(f) + \epsilon] \leq 8 \left(\frac{me}{d}\right)^d e^{-m\epsilon^2/32}$$

which holds whenever  $|D| = N \geq d$ . It follows that with probability  $1 - \delta$  over the samples, we have for all  $f \in \mathcal{C}_T$

$$\text{err}(f) \leq \widehat{\text{err}}(f) + O\left(\sqrt{\frac{d \ln(N/d) + \ln(1/\delta)}{N}}\right) \quad (15)$$

Since we choose  $T = \frac{4G_\infty^2 \ln 2N}{\epsilon^2}$ , by Theorem 1, we have

$$\forall x \in D : \|F_t - g\|_1 \leq G_\infty \sqrt{\frac{\ln 2N}{T}} \leq \frac{\epsilon}{2}$$

Since  $g$  has  $\epsilon$  margin on data with probability  $\phi$ , we have

$$\widehat{\text{err}}(F_t) \leq \widehat{\text{err}}(g) + \phi \quad (16)$$

Combining eqs. (15) and (16), we get

$$\text{err}(F_T) \leq \widehat{\text{err}}(g) + O\left(\sqrt{\frac{d \ln(N/d) + \ln(1/\delta)}{N}}\right) + \phi$$

which completes the proof.  $\square$

## C DATASET INFORMATION AND TRAINING RECEPIES

We use four publicly available real world datasets in our experiments. The train-test splits for all the dataset as well as the sources are listed here:

Dataset	Train-samples	Test-samples	Source
CIFAR-10	50000	10000	Krizhevsky et al.
SVHN-10	73257	26032	Netzer et al. (2011)
Google-13	52886	6835	Warden (2018)
DSA-19	6800	2280	Dua & Graff (2017)

We use two synthetic datasets in our experiments, *ellipsoid* and *cube*. To construct the ellipsoid dataset, we first sample a  $32 \times 32$  matrix  $B$ , each entry sampled *iid*. We define  $A := B^T B$  as our positive semi-definite matrix, and  $I[x^T A x \geq 0]$  determines the label of a data point  $x$ . We sample 10k points uniform randomly from  $[-1, 1]^{32}$  and determines their labels to construct our data sample. We randomly construct a 80-20 train-test split for our experiments.

To construct *cube*, we first sample 16 points uniform randomly from  $[-1, 1]^{32}$  and randomly split them into 4 equal sets, say  $\{S_1, \dots, S_4\}$ . As before, we sample 10k points uniformly from  $[-1, 1]^{32}$  and determine the label  $y(x)$  of each point  $x$  as,

$$y(x) = \arg \min_i \min_{x' \in S_i} \|x - x'\|.$$

## D MODEL INFORMATION, EXPERIMENTAL DETAILS AND ADDITIONAL RESULTS

### D.1 BASE MODEL CONFIGURATION

The base class configurations used for all our experiments in Figure 2 is provided in Tables 2, 3, 4 and 5. Note that we use standard model architectures provided with Pytorch and the parameters correspond to the corresponding function arguments in the pytorch implementation.

Teacher model	Residual Blocks	Embedding dims	Strides
ResNet56	1	8	1
	2,2	8,8	1,1
	2,2	16,16	1,2
	2,2,3	16,32,64	1,2,2

Table 2: Base model configuration used for ResNet20 and ResNet56 distillation on CIFAR-10.

Teacher model	Blocks	growth-rate
DenseNet121	4, 8	12
	4, 8, 8	6
	8, 16, 12	6

Table 3: Configuration used for DenseNet121 distillation on CIFAR-100.

Teacher model	hid. dims.
LSTM128	4,4
	16,8
	20,12
	20,32

Table 4: Configuration used for LSTM128 distillation on Google-13.

Teacher model	hid. dims.
GRU32	4,4
	8,16
	16,16
	32,16

Table 5: Configuration used for GRU32 distillation on DSA-19.

### D.2 TRAINING RECEPIES

We use stochastic gradient descent (SGD) with momentum for all our experiments. For experiments on CIFAR100 and CIFAR10, we use a learning rate of 0.1, a momentum parameter of 0.9, and weight decay of  $5 \times 10^{-4}$ . We train for 200 epochs and reduce the learning rate by a factor of 0.2 in after 30%, 60% and 90% of the epoch execution. For experiments with time series data, Google-13 and DSA-19, we use a fixed learning rate of 0.05 and a momentum of 0.9. We do not use weight decay or learning rate scheduling for time-series data.

### D.3 ADDITIONAL RESULTS

**FLOPS measurement.** To measure the total floating point operations required for inference, we use the *Deep Speed* framework [Rasley et al. \(2020\)](#). We randomly initialize a single sample with batch-size set to 1, and profile it to obtain our values for FLOPs for all real-world data sets.

**Connections, parallelization and execution schemes.** Our focus in this work has been sequential execution of the models. While reusing previously computed features is clearly beneficial for finding weak learners in this setup, the presence of connections across models prevent them from being scheduled together for execution whenever otherwise possible. To manage this trade off between parallelization and expressivity, we try to restrict the number of connections to at most one between models, and further restrict the connection to later layers of the model. Connections in the later layers of networks impose fewer sequential blocks in inference and allows for better parallelization.

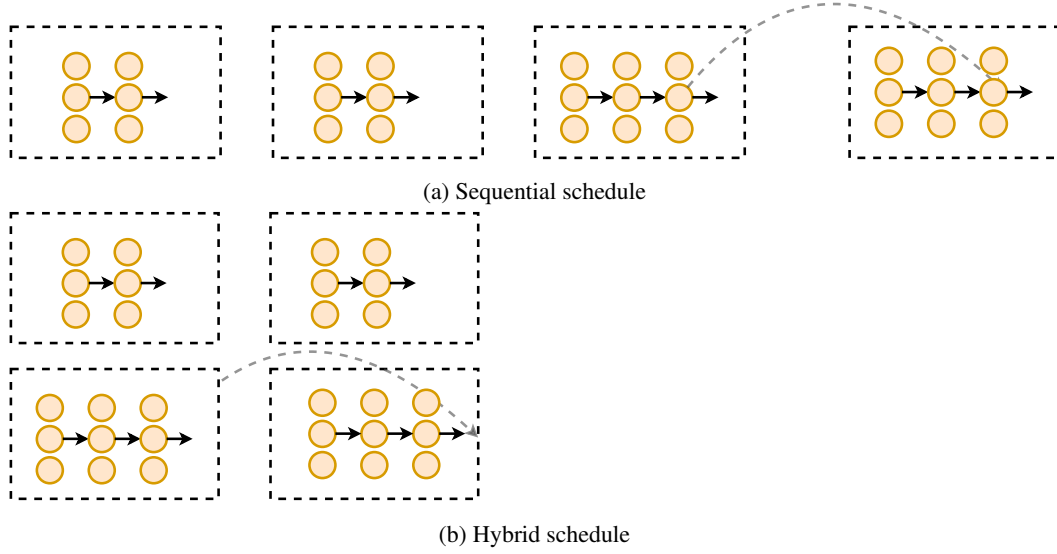


Figure 4: A schematic description of a sequential execution scheme for an ensemble of four models, being evaluated one after the other from left to right. The last model in the ensemble reuses the actions of the previous one, causing a blocking dependency. Thus we cannot trivially execute all models in parallel. However, since the connection is between the last layers of the network, we can construct hybrid execution schemes as in (b). Here, pairs of models are executed in together.

Let  $\phi_{t,l}(x)$  denote the activation produced at layer  $l$  by the weak learner at round  $t$ , on data-point  $x$ . Then some of the connections we consider are

- $\phi_{(t,l)}(x) - \phi_{(t+1,l)}(x)$ , to learn the error in features at layer  $l$ .
- $\phi_{(t,l)}(x) + \text{id}(x)$ , standard residual connection at layer  $l$ .
- $\phi_{(t+1,l)}[\phi_{(1,l)}(x), \dots, \phi_{(t,l)}(x)]$ , dense connections at layer  $l$  across rounds.
- Simple accumulation operations.
- Recurrent networks: LSTM and GRU.