Friedrich-Alexander-Universität
Technische Fakultät

Advanced Deep Learning

Tricks of the Trade

K. Breininger, V. Christlein

Artificial Intelligence in Medical Imaging + Pattern Recognition Lab,

Friedrich-Alexander-Universität Erlangen-Nürnberg SoSe 2023

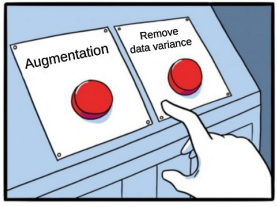
Several hints from: Aline Sindel, Florian Kordon, Florian Thamm, Martin Mayr, Mathias Seuret
Knowledge distillation slides adapted from Marco Schnell's Seminar slides

1. General
2. Training++
3. Speed++
3.1 Knowledge Distillation
3.2 Architecture Changes
4. Performance++
4.1 Techniques
4.2 A recipe for getting started
4.3 Challenge Winning Strategies

Data

Visualization

- Visualize your data → understand it and get a feeling for it
- Which invariances are needed?
  1. Augmentation
  2. Input homogenization via pre-processing



Dataset splits

Split your dataset **properly** (e.g., document/patient-independent) in three **disjoint** sets:

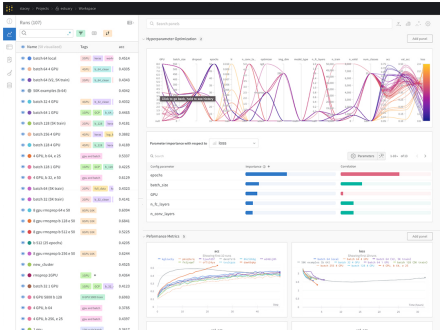
- Train: for NN update
- Validation (development): for hyper-parameter tuning
- Test: for final evaluation

Source: https://imgflip.com/memegenerator/Two-Buttons

Error Analysis

Monitoring

- Monitor training and validation loss/other metrics + examples:
  - Older: Tensorboard or Sacred
  - Weights & biases (`https://wandb.ai`)
  - Neptune (`https://neptune.ai`)
  - Comet (`https://www.comet.com`)
- Use callbacks ➔ omit main module changes
- Save your visualizations as vector graphics if possible
- Visualize input/output of first batch (training+validation) to ensure correctness of data loading w. augmentations & predictions



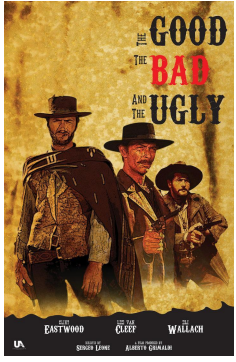
Source: `https://wandb.ai`

Error Analysis

- Look at both training and validation error
  - Difference between training and validation ($\approx$ variance)
- Do both quantitative & qualitative evaluation
  - study data points that are at the opposing sides of the error spectrum
  - try to characterize them (the good, the bad, the ugly)

→ Identify commonalities, e.g., image characteristic that always leads to high errors

- Create confusion matrices



Source: https://www.filmaffinity.com/us/movieimage.php?imageId=730766925

Reproducibility

- Comment your code, esp. mark what is taken from somewhere (source)
- Write test cases if modules and functions are plausible
- Provide all steps to run the code (requirements.txt etc.)
- Store numerical results in pandas dataframes
- → Fast iteration over different visualizations
- Use git!

Source: https://s3.amazonaws.com/msia423/memes/60-percent-of-time-deployments-work-everytime.jpg

Reproducibility++

- Documentation, documentation, documentation
  ➔ automatize as much as possible
- Experiment management, e.g., Hydra, Optuna
  - Recording & tracking of hyper-parameters
  - Advanced hyper-parameter search (beyond grid search)
  - Can be combined with W&B, MLflow, neptune.ai, ...

https://hydra.cc/

https://optuna.org/

Evaluation Metric

Choose evaluation metric wisely

Example: Segmentation

- Typically overlap metrics used IoU/DICE score
- Sometimes more interested in contour: Hausdorff distance or average symmetric surface distance

→ Find single metric that really tracks what you are trying to achieve

General strategy

- Find simple, relatively fast, relatively low-resource-consumption configuration that obtains "reasonable" result.

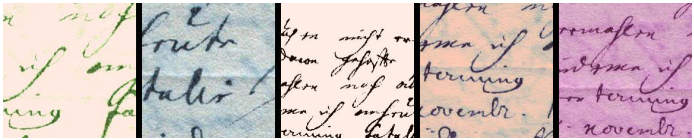
→ Stick to basic, established architectures first in the respective domains:

  - e.g., ResNet50 for first prototype
  - or ViT (note: typically more difficult to train)
  - later for a pure CNN approach: ConvNeXt as one of the best archs

- Overfit on small subset first
- Look at dev set examples to evaluate ideas
- If there is no suitable "toy dataset" → Build one

1. General
2. Training++
3. Speed++
3.1 Knowledge Distillation
3.2 Architecture Changes
4. Performance++
4.1 Techniques
4.2 A recipe for getting started
4.3 Challenge Winning Strategies

Augmentation



- Has to fit to your problem
- Use established ones if possible, e.g., image-based augmentations of timm library: `https://github.com/huggingface/pytorch-image-models`
- Auto-augment methods typically expensive
- Default PyTorch augmentations might be slow, try other libraries:
  - → Kornia, Albumentations, imgaug, TorMentor, Augmentor, OpenCV
- Strategy: offline or online?
  - Trade-off between flexibility and data storage
  - Keep parallelization in mind

Augmentations (cont.)

Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4

- Class-imbalance? → choose higher weight for less frequent class (Remix [14])
- More augmentation techniques? see e.g. [26]

Source: Yun, Han, Chun, et al. 2019 [12]

Data Handling

- Save large data chunk
- Use compressed format
- Don't use 'HDF5' (single-threaded, random access difficult, no parallel trafos possible)
  - → 'pkl.gz' (other compression algo possible)
  - → 'npz' files
  - → New: deeplake (https://github.com/activeloopai/deeplake)
  - Additional suggestions by students for compression algorithms LZ4, zstd
- Always use fastest memory: GPU RAM > CPU RAM > SSD > HDD > external mount
- → Load large data chunks into CPU RAM, then smaller into GPU RAM
- Leave checkpoints in RAM, only write it from time-to-time to hard-drive
  - Keep wall-time into account
  - Can also be done in another thread

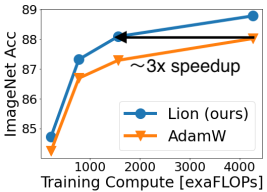
https://github.com/lz4/lz4

https://github.com/facebook/zstd

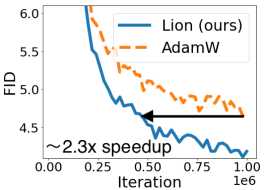
Optimizer

- Standard: (improved) ADAM family: AdamW, RAdam
- Alternatives: novograd [18], adafactor [19], LARS [28], LAMB [29]
- Newest: Lion [1]: automatically discovered optimizer

Classification



Generation



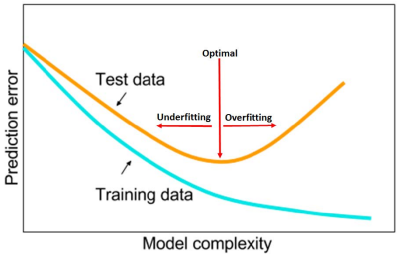
Program: Lion: $\beta_1 = 0.9$ and $\beta_2 = 0.99$

```
def train(weight, gradient, momentum, lr):
  update = interp(gradient, momentum, β1)
  update = sign(update)
  momentum = interp(gradient, momentum, β2)
  weight_decay = weight * λ
  update = update + weight_decay
  update = update * lr
  return update, momentum
```

Source: Chen, Liang, Huang, et al. 2023 [1]

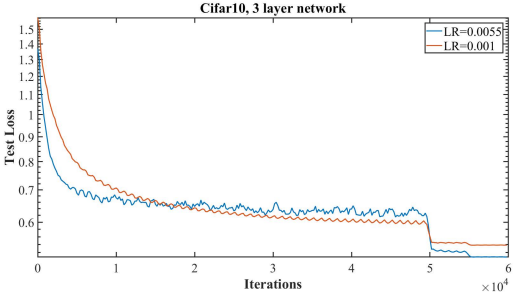
Learning rate tracking & diagnosing

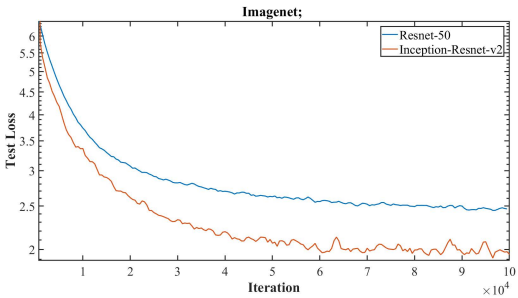
- "Well begun is half done"
- Underfitting vs. overfitting trade-off
- Examine train & validation loss [30]



Source: Smith 2018 [30]

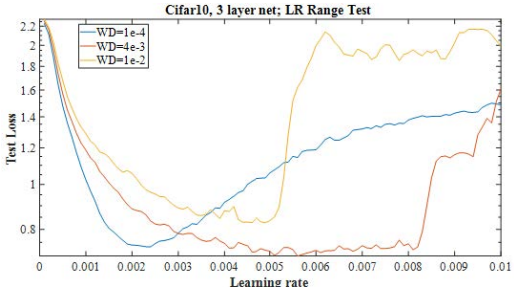
Learning rate tracking & diagnosing - underfitting

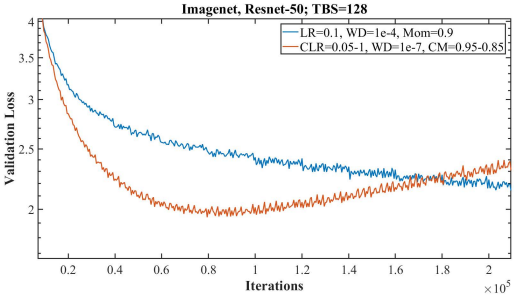




Source: Smith 2018 [30]

Learning rate tracking & diagnosing - overfitting

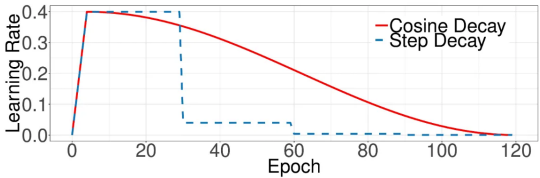




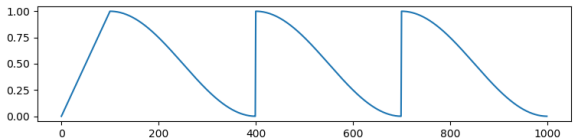
Source: Smith 2018 [30]

Scheduler

- Most popular: cosine scheduler [20]

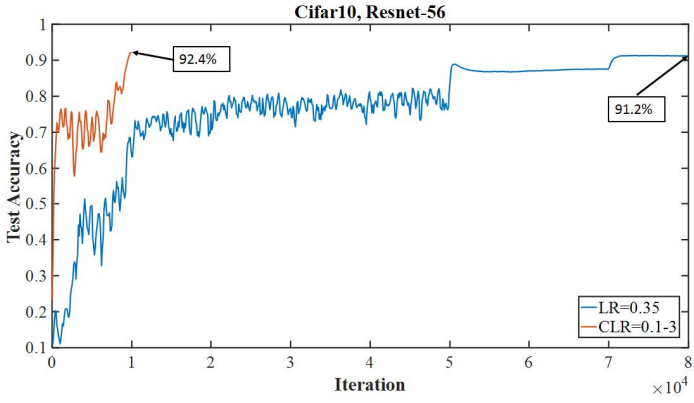


- Cosine LR scheduler with restarts, good esp. if method susceptible to initial LR



Sources:
https://pub.towardsai.net/are-you-sure-that-you-can-implement-image-classification-networks-d5f0bffb242d
https://huggingface.co/docs/transformers/main_classes/optimizer_schedules

Superconvergence

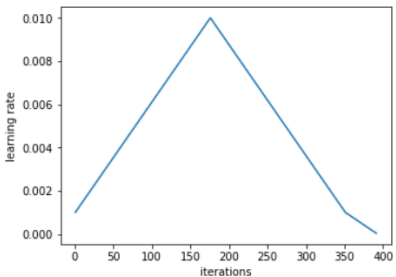


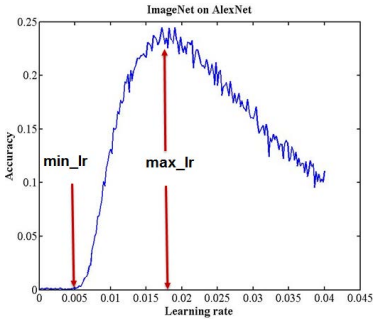
Source: Smith17-1Cycle

Cyclic learning rates - 1cycle

1cycle policy for “super-convergence” [23]:

- LR range test: pretraining run (few epochs) to determine max. learning rate
- Linear increase to max (warmup) and linear decrease + annealing





Source: Smith17-1Cycle

Hyper-parameter Tuning

- Critical performance factor
- Grid-search → gives intuition
- And/or random search
  - → better exploration of search space
  - → good results at reasonable time
- Use libraries: optuna, hyper-opt

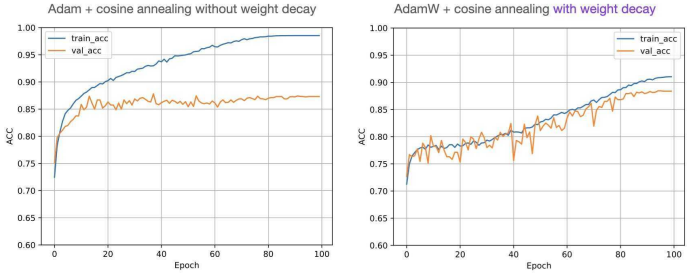
→ Bayesian hyper-parameter optimization

Source: https://twitter.com/rasbt/status/1637930578983919616?t=tlabVaJTvUjtvI-VdVb9cg&s=19

Hyper-parameter Tuning (cont.)

Most important parameters:

- Initial learning rate
- Weight decay
- Note: batch size should not be treated as a tunable hyper-parameter [2]



Source: https://twitter.com/rasbt/status/1637930578983919616?t=tlabVaJTvUjtvI-VdVb9cg&s=19

Tuning Strategy [2]

Repeat:

1. Identify appropriately-scoped goal for next experiments
2. Design and run set of experiments that makes progress towards this goal
3. Learn what you can from the results
   - Exploration $\gg$ exploitation
   - Prioritizing insight over short term gains
4. Consider whether to launch the new best configuration

More details: Tuning Playbook by Godbole, Dahl, Gilmer, et al. 2023 [2]

1. General
2. Training++
3. Speed++
3.1 Knowledge Distillation
3.2 Architecture Changes
4. Performance++
4.1 Techniques
4.2 A recipe for getting started
4.3 Challenge Winning Strategies

Knowledge Distillation

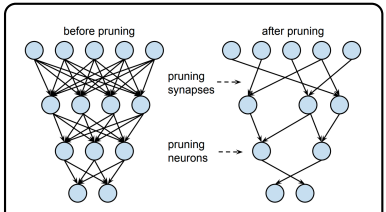
Motivation

Why is decreasing the size necessary? [6], [7]

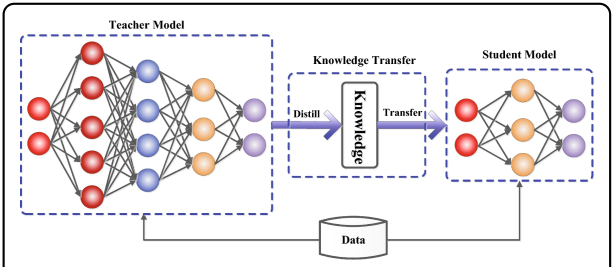
- Not deployable
- Computational amount
- Energy consumption
- Time effort

Sources:
https://komputer.dk/computer/usa-bygger-verdens-hurtigste-computer
https://m.media-amazon.com/images/I/71iIaQjJunL._AC_SX522_.jpg

Approaches



Neural Network Pruning [9]



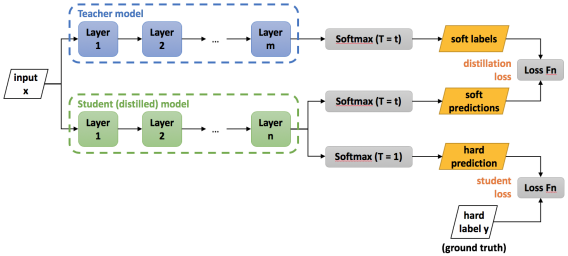
Knowledge Distillation [8]

Sources:
left: https://miro.medium.com/max/1400/1*nicFUkeUpWMW1w_hUVtZiw.png
right: Gou et al. 21 [5]

Distillation Loss

Goal: Minimize differences between teacher-student output → additional distillation loss:

$$\mathcal{L} = \sum_{x_i \in \mathcal{X}} \alpha \, \mathcal{L}_{\text{CE}}(y, f_S(x_i)) + \beta \, \mathcal{L}_{\text{KD}}(f_S(x_i), f_T(x_i))$$



E.g. [8]:

$$\mathcal{L}_{\text{KD}} = \text{KL}\Big(\text{softmax}\big(f_T(x_i)/\tau\big), \text{softmax}\big(f_S(x_i)/\tau\big)\Big)$$

f_T, f_S: teacher, student model

τ: temperature parameter

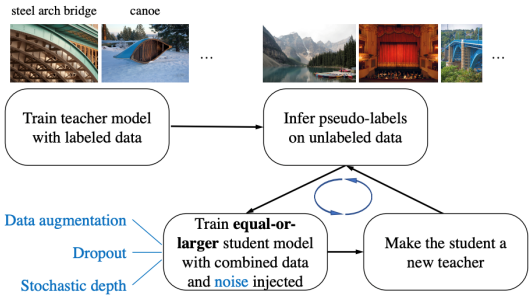
Source: https://intellabs.github.io/distiller/knowledge_distillation.html

KD for Improved Accuracy

Goal: Increase performance → not necessarily decrease model

Noisy Student Training [10]

- Semi-supervised learning (uses additional unlabeled data)
- Student training with additional model/input noise
  - Data augmentation
  - Dropout
  - Stochastic depth
- Data filtering: filter out low-confidence images from teacher
- Soft and hard pseudo-labels



Source: Xie, Luong, Hovy, et al. 2020 [10]

Architecture Changes

Architecture Speed-Ups

ReZero [3]

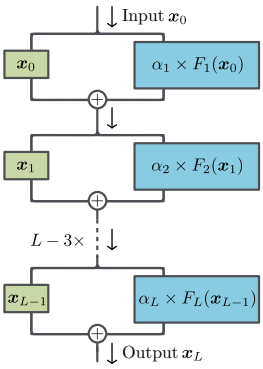
(1) Deep Network	$\boldsymbol{x}_{i+1} = F(\boldsymbol{x}_i)$
(2) Residual Network	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + F(\boldsymbol{x}_i)$
(3) Deep Network + Norm	$\boldsymbol{x}_{i+1} = \text{Norm}\big(F(\boldsymbol{x}_i)\big)$
(4) Residual Network + Pre-Norm	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + F(\text{Norm}\big(\boldsymbol{x}_i\big))$
(5) Residual Network + Post-Norm	$\boldsymbol{x}_{i+1} = \text{Norm}\big(\boldsymbol{x}_i + F(\boldsymbol{x}_i)\big)$
(6) **ReZero**	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i F(\boldsymbol{x}_i)$

- For each layer: residiual connection with learnable parameter α
- Initialize $\alpha = 0$ → Initializes to identity → "dynamic isometry" (singular values of network's Jacobian ≈ 1)

→ Enables training of 10 000 FC layers and 100 Transformer layers

→ Converges 56–85 % speedup

→ Mostly, slightly behind SOTA



Source: Bachlechner, Majumder, Mao, et al. 2021 [3]

Architecture Speed-Ups

Notes

Interesting but not straight-forward papers that achieve ResNet performance and speed without skip connections by carefully controlling the “shape" of the network’s initialization-time kernel function:

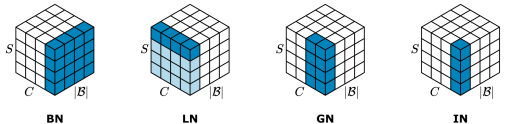
- Deep Kernel shaping (DKS) [21]
- Extension: Tailored Activation Transformation (TAT) [22]

☞ Implementation: `https://github.com/deepmind/dks`

Normalization in NNs - Batch Normalization

Observations when using Batch Normalization (BN)

+ Higher learning rates
+ Increases stability
+ Less sensitive to initialization



➔ Recommendation: **Instance Norm**

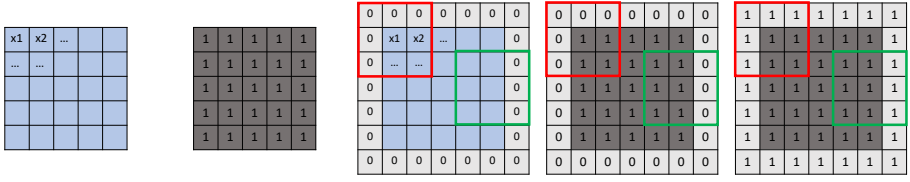
- Does not work well with small batch sizes
- Performs poorly for dependent/correlated samples
- **Training vs. inference statistics**
- Does not work well with other regularization techniques
- Computational overhead

Source: https://iclr-blog-track.github.io/2022/03/25/unnormalized-resnets/, who adapted it from [32]

1. General
2. Training++
3. Speed++
3.1 Knowledge Distillation
3.2 Architecture Changes
4. Performance++
4.1 Techniques
4.2 A recipe for getting started
4.3 Challenge Winning Strategies

Techniques

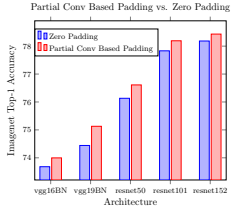
Partial Convolutions [25]



- Fix of zero-padding:

$$x'_{(i,j)} = \mathbf{W}^T \mathbf{X}^{p0}_{(i,j)} r_{(i,j)} + b \quad \text{with} \quad r_{(i,j)} = \frac{||\mathbf{1}^{p1}_{(i,j)}||_1}{||\mathbf{1}^{p0}_{(i,j)}||_1}$$

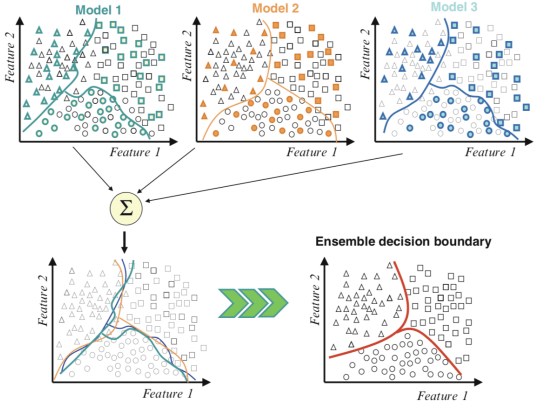
- Effect rather small (larger for inpainting [24])



Source: Liu, Shih, Wang, et al. 2018 [25]

Ensembling

- Can drastivally improve results
- Ensemble of
  - Different model outputs
  - Same model outputs
- Possible to weight them differently, e.g., by validation performance



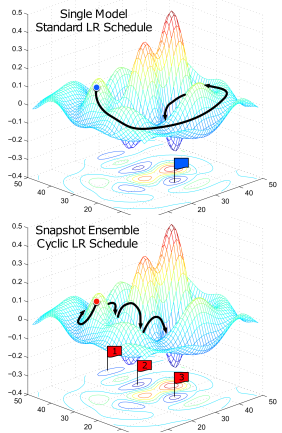
Source: https://machinelearningmastery.com/how-ensemble-learning-works/

Stochastic Weight Averaging [4]

- Uses a cyclic learning rate
- Two models:
  running average (θ_{SWA} and **"normal"** θ)

$$\theta_{SWA} \leftarrow \frac{\theta_{\text{SWA}} \cdot m + \theta}{m + 1}$$

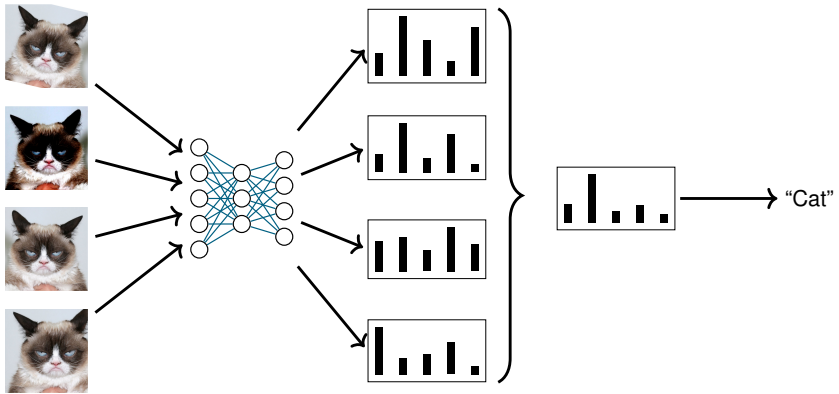
→ **Single model as ensemble**



Source: Huang, Li, Pleiss, et al. 2017 [4]

Test-time Augmentation

Augment original image → run all versions through the model → aggregate results (→ compute uncertainty)



Source: https://amva4newphysics.wordpress.com/2018/04/26/train-time-test-time-data-augmentation/

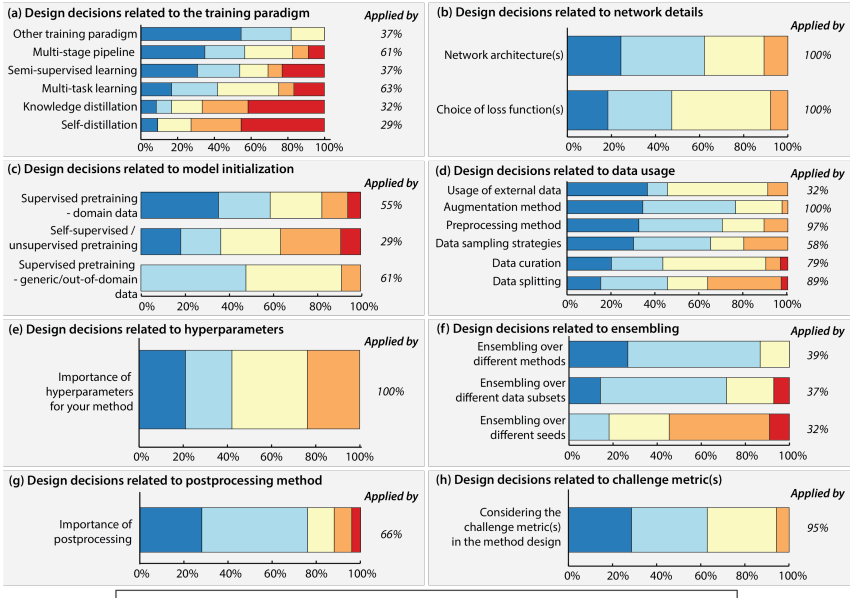
A recipe for getting started

A recipe for getting started

Following Karpathy's blog and Google's tuning playbook:

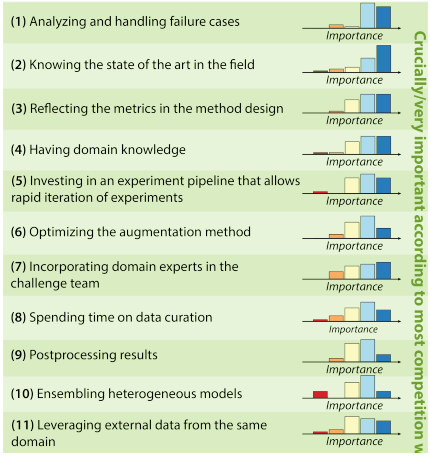
1. Become one with the data
2. Choose a well-established architecture
3. Set up end-to-end evaluation & get dump baselines

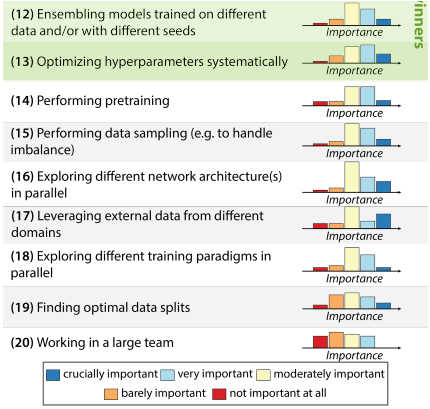
Challenge Winning Strategies



Source: Eisenmann, Reinke, Weru, et al. 2023 [27]

Successful Strategies





Source: Eisenmann, Reinke, Weru, et al. 2023 [27]

Winner Profile

Team composition and expertise

- **Affiliation:** Academia (86%)
- **Highest degree:** Doctorate degree (41%)
- **Background:** Computer science (55%)
- **Experience in medical image analysis:** 5 years
- **Experience in deep learning:** 4 years
- **Prior participation in a competition:** No (50%)
- **Team size:** 3
- **Involvement of domain experts:** No (82%)

Infrastructure and tools

- **GPU-based development:** Yes (95%)
- **Programming language:** Python (95%)
- **Library for method development:** PyTorch (86%)
- **Total amount of GPU hours:** 490
- **Type of GPU resources:** Patchwork cluster of workstations/ compute nodes (33%)
- **Infrastructure bottleneck:** No (70%)

Strategy

- **Incentive for participation:** Knowledge exchange (73%)
- **Decision to submit:** 3 weeks before submission
- **Hours spent before deciding to submit:** 120
- **Working hours for method development:** 80
- **Hyperparameter tuning:** Human-driven (50%)
- **Enough time for development:** Yes (86%)
- **Max. time spent on:** Model design and implementation (25%)
- **Believes in potential to win:** Yes (45%)
- **Recommends:** Strategies 1-13 in Fig 4

Source: Eisenmann, Reinke, Weru, et al. 2023 [27]

Summary

- Training NNs is an art

→ Involves a lot of engineering

- SOTA assumes many tricks
- Document your code and make it reproducible safes a lot of trouble

☞ Don't brute-force

→ Try to **understand** the data, the problem, the important tuning factors, etc.

Not discussed but also interesting: Mixed precision training, multi-processing, normalization, ...

NEXT TIME

ADVANCED

ON DEEP LEARNING

Ethics in Machine Learning

Recommended Reading: https://plato.stanford.edu/entries/ethics-ai/

Source: https://plato.stanford.edu/entries/ethics-ai/

Comprehensive Questions

- Why does it not make sense to (over-)tune hyperparameters and esp. batch size on your validation set?
- What augmentation techniques are there and how can you use them at test time?
- Which optimizers and learning rate schedulers would you use?
- How does knowledge distillation work?
- How does ReZero work?
- How does partial convolution work?

Further Reading

- Deep Learning Tuning Playbook [2]: https://github.com/google-research/tuning_playbook
- Pytorch Performance Tuning Guide: https://pytorch.org/tutorials/recipes/recipes/tuning_guide.html
- Andrew Ng: Machine Learning Yearning Book https://info.deeplearning.ai/machine-learning-yearning-book
- Most of the works by Leslie N. Smith:
  - List of arxiv articles: https://arxiv.org/search/cs?searchtype=author&query=Smith%2C+L+N
  - e.g. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay (link)
- ResNet strikes back: An improved training procedure in timm [31]

References

References I

[1] Xiangning Chen, Chen Liang, Da Huang, et al. Symbolic Discovery of Optimization Algorithms. 2023. arXiv: 2302.06675 [cs.LG].

[2] Varun Godbole, George E. Dahl, Justin Gilmer, et al. Deep Learning Tuning Playbook. Version 1.0. 2023.

[3] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, et al. "ReZero is all you need: fast convergence at large depth". In: Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence. Vol. 161. Proceedings of Machine Learning Research. PMLR, 27–30 Jul 2021, pp. 1352–1361.

[4] Gao Huang, Yixuan Li, Geoff Pleiss, et al. "Snapshot Ensembles: Train 1, get M for free". In: CoRR abs/1704.00109 (2017). arXiv: 1704.00109.

[5] Jianping Gou, Baosheng Yu, Stephen J. Maybank, et al. "Knowledge Distillation: A Survey". In: International Journal of Computer Vision 129.6 (2021), pp. 1789–1819.

References II

[6] Lin Wang and Kuk-Jin Yoon. “Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks”. In: IEEE transactions on pattern analysis and machine intelligence 44.6 (2022), pp. 3048–3068.

[7] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, et al. “Improved Knowledge Distillation via Teacher Assistant”. In: AAAI Conference on Artificial Intelligence 34.04 (2020), pp. 5191–5198.

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the Knowledge in a Neural Network”. In: arXiv e-prints abs/1503.02531 (2015).

[9] Jonathan Frankle and Michael Carbin. “The Lottery Ticket Hypothesis: Training Pruned Neural Networks”. In: CoRR abs/1803.03635 (2018). arXiv: 1803.03635.

[10] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, et al. “Self-Training With Noisy Student Improves ImageNet Classification”. In: (June 2020), pp. 10684–10695.

References III

[11] Terrance DeVries and Graham W. Taylor.
Improved Regularization of Convolutional Neural Networks with Cutout. 2017. arXiv: 1708.04552 [cs.CV].

[12] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, et al. “CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features”. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). Oct. 2019, pp. 6022–6031.

[13] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, et al. “mixup: Beyond Empirical Risk Minimization”. In: International Conference on Learning Representations. 2018.

[14] Hsin-Ping Chou, Shih-Chieh Chang, Jia-Yu Pan, et al. “Remix: Rebalanced Mixup”. In: Computer Vision – ECCV 2020 Workshops. Cham: Springer International Publishing, 2020, pp. 95–110.

[15] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, et al. “SMOTE: synthetic minority over-sampling technique”. In: Journal of artificial intelligence research 16 (2002), pp. 321–357.

References IV

[16] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. “Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning”. In: Advances in Intelligent Computing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 878–887.

[17] Haibo He, Yang Bai, Edwardo A. Garcia, et al. “ADASYN: Adaptive synthetic sampling approach for imbalanced learning”. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computation June 2008, pp. 1322–1328.

[18] Boris Ginsburg, Patrice Castonguay, Oleksii Hrinchuk, et al. Stochastic Gradient Methods with Layer-wise Adaptive Moments for Training of Deep Networks. 2020. arXiv: 1905.11286 [cs.LG].

[19] Noam Shazeer and Mitchell Stern. “Adafactor: Adaptive Learning Rates with Sublinear Memory Cost”. In: Proceedings of the 35th International Conference on Machine Learning. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 4596–4604.

References V

[20] Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: International Conference on Learning Representations. 2017.

[21] James Martens, Andy Ballard, Guillaume Desjardins, et al. Rapid training of deep neural networks without skip connections or normalization layers using Deep K 2021. arXiv: 2110.01765 [cs.LG].

[22] Guodong Zhang, Aleksandar Botev, and James Martens. “Deep Learning without Shortcuts: Shaping the Kernel with Tailored Rectifiers”. In: International Conference on Learning Representations. 2022.

[23] Leslie N. Smith and Nicholay Topin. “Super-Convergence: Very Fast Training of Residual Networks Using Large Learning Rates”. In: CoRR abs/1708.07120 (2017). arXiv: 1708.07120.

[24] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, et al. “Image Inpainting for Irregular Holes Using Partial Convolutions”. In: Computer Vision – ECCV 2018. Cham: Springer International Publishing, 2018, pp. 89–105.

References VI

[25] Guilin Liu, Kevin J. Shih, Ting-Chun Wang, et al. Partial Convolution based Padding. 2018. arXiv: 1811.11718 [cs.CV].

[26] Mingle Xu, Sook Yoon, Alvaro Fuentes, et al. “A Comprehensive Survey of Image Augmentation Techniques for Deep Learning”. In: Pattern Recognition 137 (2023), p. 109347.

[27] Matthias Eisenmann, Annika Reinke, Vivienn Weru, et al. “Why Is the Winner the Best?” In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). June 2023, pp. 19955–19966.

[28] Yang You, Zhao Zhang, Cho-Jui Hsieh, et al. “ImageNet Training in Minutes”. In: Proceedings of the 47th International Conference on Parallel Processing. ICPP ’18. Eugene, OR, USA: Association for Computing Machinery, 2018.

[29] Yang You, Jing Li, Sashank Reddi, et al. “Large Batch Optimization for Deep Learning: Training BERT in 76 minutes”. In: International Conference on Learning Representations. 2020.

References VII

[30] Leslie N. Smith. “A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay”. In: CoRR abs/1803.09820 (2018). arXiv: 1803.09820.

[31] Ross Wightman, Hugo Touvron, and Hervé Jégou. ResNet strikes back: An improved training procedure in timm. 2021. arXiv: 2110.00476 [cs.CV].

[32] Yuxin Wu and Kaiming He. “Group Normalization”. In: Computer Vision – ECCV 2018. Cham: Springer International Publishing, 2018, pp. 3–19.