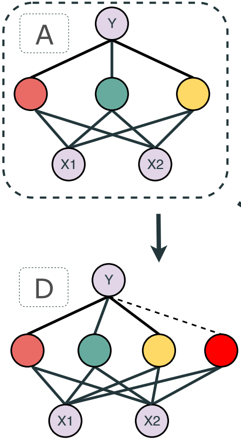
Knowledge Transfer with Jacobian Matching

Suraj Srinivas & François Fleuret

Machine Learning group
Idiap Research Institute & EPFL



- **Different parameterizations** can represent the **same function**
- **Parameterization-invariant** tools describe the function
- **Regularize the function**, not its parameterization

INPUT JACOBIAN OF NEURAL NETWORKS

$$\nabla_x y = \begin{bmatrix} \frac{\partial y}{\partial x_0} & \frac{\partial y}{\partial x_1} & \cdots & \frac{\partial y}{\partial x_D} \end{bmatrix}$$

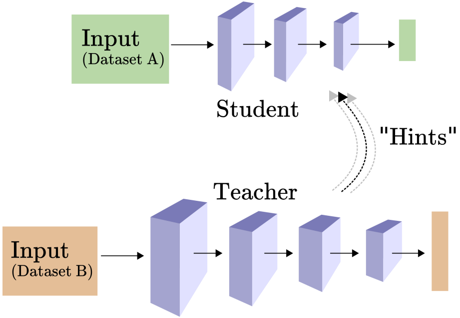
- In general, for input $\in \mathbb{R}^D$ and output $\in \mathbb{R}^K$, the Jacobian $\in \mathbb{R}^{D \times K}$
- Jacobian is **invariant to parameterization** of the function

INPUT JACOBIAN OF NEURAL NETWORKS

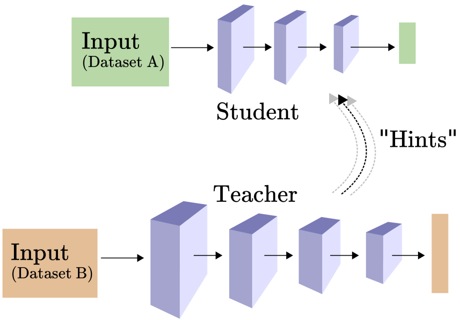
$$\nabla_x y = \begin{bmatrix} \frac{\partial y}{\partial x_0} & \frac{\partial y}{\partial x_1} & \cdots & \frac{\partial y}{\partial x_D} \end{bmatrix}$$

- In general, for input $\in \mathbb{R}^D$ and output $\in \mathbb{R}^K$, the Jacobian $\in \mathbb{R}^{D \times K}$
- Jacobian is **invariant to parameterization** of the function
- For ReLU nets without bias, $y = \nabla_x y^T x$

KNOWLEDGE TRANSFER BETWEEN NEURAL NETS



- If datasets A == B, task = distillation; else task = transfer learning
- If architectures of both nets are same, we can **copy weights**
- 'Hints' must be **parameterization invariant**



- If datasets A == B, task = distillation; else task = transfer learning
- If architectures of both nets are same, we can **copy weights**
- ‘Hints’ must be **parameterization invariant**
- Czarnecki et al. [2017] and Zagoruyko and Komodakis [2017] previously used Jacobians, but did not motivate choice of loss function

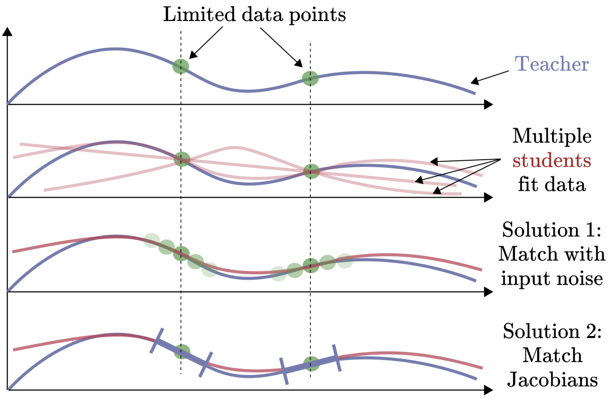


Figure: Teacher-student learning in a simple 1D case.

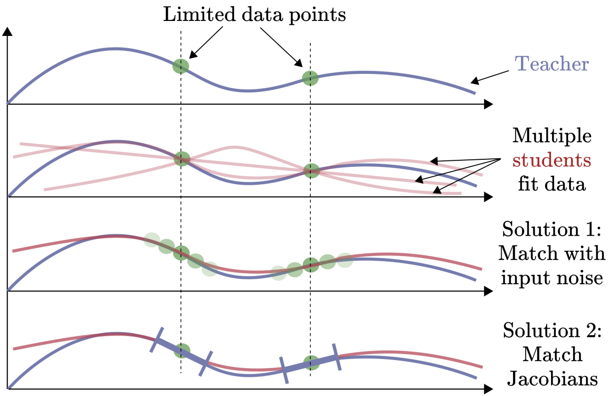


Figure: Teacher-student learning in a simple 1D case.

$$\underbrace{\mathbb{E}_{\boldsymbol{\xi}}\left[(\mathcal{T}(\mathsf{x}+\boldsymbol{\xi})-\mathcal{S}(\mathsf{x}+\boldsymbol{\xi})\right]^2}_{\text{Matching outputs with input noise}} = \underbrace{(\mathcal{T}(\mathsf{x})-\mathcal{S}(\mathsf{x}))^2}_{\text{Matching outputs}} + \sigma^2 \underbrace{\|\nabla_{\mathsf{x}}\mathcal{T}(\mathsf{x})-\nabla_{\mathsf{x}}\mathcal{S}(\mathsf{x})\|_2^2}_{\text{Matching Jacobians}}$$

JACOBIAN NORM REGULARIZATION

$$\overbrace{\mathbb{E}_{\boldsymbol{\xi}}\left[\mathsf{y}(\mathsf{x}) - \mathcal{S}(\mathsf{x}+\boldsymbol{\xi})\right]^2}^{\text{Matching with input noise}} = \overbrace{\left(\mathsf{y}(\mathsf{x}) - \mathcal{S}(\mathsf{x})\right)^2}^{\text{Matching outputs}} + \sigma^2 \overbrace{\left\|\nabla_{\mathsf{x}}\mathcal{S}(\mathsf{x})\right\|_2^2}^{\text{Jacobian norm}}$$

$$\overbrace{\mathbb{E}_{\boldsymbol{\xi}}\left[\mathsf{y}(\mathsf{x})-\mathcal{S}(\mathsf{x}+\boldsymbol{\xi})\right]^2}^{\text{Matching with input noise}} = \overbrace{\left(\mathsf{y}(\mathsf{x})-\mathcal{S}(\mathsf{x})\right)^2}^{\text{Matching outputs}} + \sigma^2 \overbrace{\|\nabla_{\mathsf{x}}\mathcal{S}(\mathsf{x})\|_2^2}^{\text{Jacobian norm}}$$

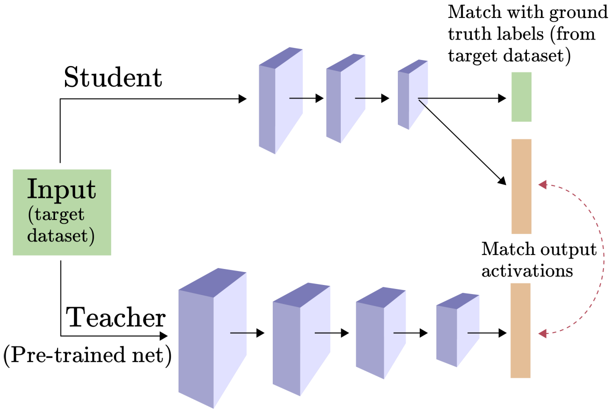
- First described by Bishop [1995]

JACOBIAN NORM REGULARIZATION

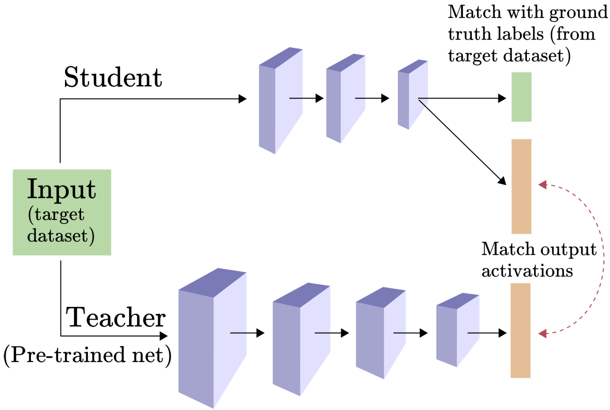
$$\overbrace{\mathbb{E}_{\boldsymbol{\xi}}\left[\mathsf{y}(\mathsf{x}) - \mathcal{S}(\mathsf{x} + \boldsymbol{\xi})\right]^2}^{\text{Matching with input noise}} = \overbrace{\left(\mathsf{y}(\mathsf{x}) - \mathcal{S}(\mathsf{x})\right)^2}^{\text{Matching outputs}} + \sigma^2 \overbrace{\left\|\nabla_{\mathsf{x}}\mathcal{S}(\mathsf{x})\right\|_2^2}^{\text{Jacobian norm}}$$

- First described by Bishop [1995]
- For linear models
  Jacobian norm regularizer $= \ell_2$ regularizer on weights
- For neural networks
  Jacobian norm regularizer $\neq$ layerwise ℓ_2 weight regularizer

Applying Jacobian matching to Transfer Learning



- **Multi-task objective** for the student
  - Match ground truth labels
  - Mimic teacher's response (distillation)



- **Multi-task objective** for the student
  - Match ground truth labels
  - Mimic teacher's response (distillation)
- **Important**: Teacher is **not trained** on target dataset

- Teacher is not trained on data being matched
- Improved matching may not improve transfer learning

WHY SHOULD IT WORK?

- Teacher is not trained on data being matched
- Improved matching may not improve transfer learning

- Theoretical results:
  - LwF helps nets with small Lipschitz constants, and when "distance" between source and target datasets are small
  - Jacobian matching always improves LwF

- Equivalence between Jacobian matching and training with noise is crucial to the proof

Table: Transfer Learning from Imagenet to MIT Scenes dataset measured by test accuracy (%).

# of Data points per class →	25	50	Full
No Transfer Learning	35.19	46.38	59.33
Fine-tuning Oracle [1]	57.65	64.18	71.42
LwF	45.08	55.22	65.22
LwF + Jacobians	45.26	56.49	66.04
LwF + attention	46.01	57.80	**67.24**
LwF + attention + Jacobians	**47.31**	**58.35**	**67.31**

[1] Requires teacher and student to have the same architecture

- Jacobians are a good parameterization-invariant quantity to use for distillation, transfer learning and improving robustness to random noise
- The data augmentation viewpoint of Jacobian matching motivates its use in low data settings.

Questions?

email: suraj.srinivas@idiap.ch

REFERENCES

Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M Kitani. N2n learning: Network to network compression via policy gradient reinforcement learning. ICLR, 2018.

LJ Ba and R Caruana. Do deep networks really need to be deep. Advances in neural information processing systems, 27:1–9, 2014.

Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. Neural Computation, 1995.

Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 535–541. ACM, 2006.

Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. AAAI, 2018.

Wojciech Marian Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Świrszcz, and Razvan Pascanu. Sobolev training for neural networks. NIPS, 2017.

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

Zhizhong Li and Derek Hoiem. Learning without forgetting. In European Conference on Computer Vision, pages 614–629. Springer, 2016.

J Ross Quinlan. Generating production rules from decision trees. In IJCAI, volume 87, pages 304–307. Citeseer, 1987.

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 Conference on Computer Vision and Pattern Recognition, pages 618–626, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. arXiv preprint arXiv:1703.01365, 2017.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. ICLR, 2017.

Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. arXiv preprint arXiv:1702.04595, 2017.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578, 2016.