CS7150 Deep Learning

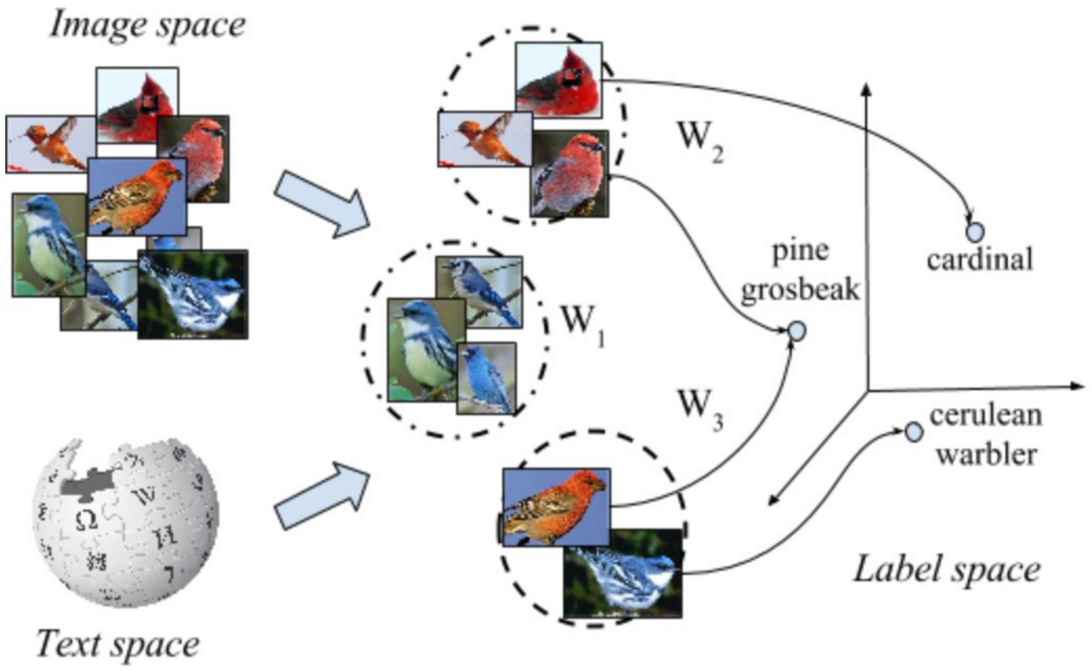
Jiaji Huang

https://jiaji-huang.github.io

03/16/2024

Recap of Last Lecture

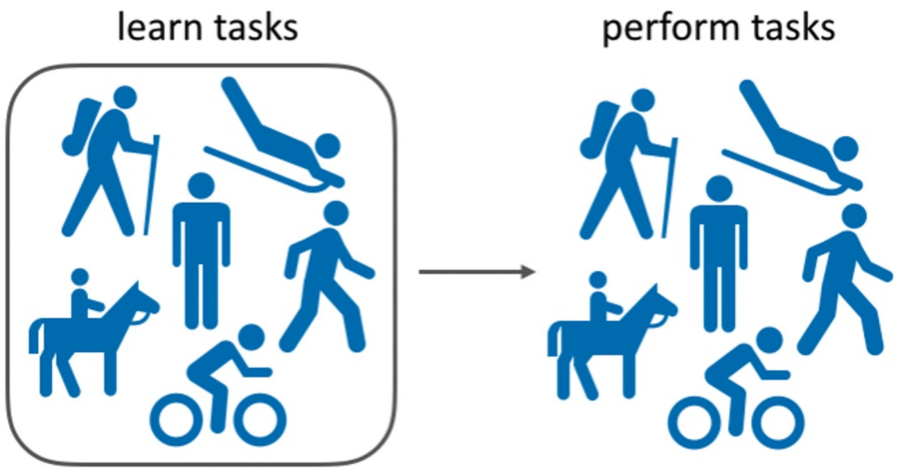
- Zero Shot Learning

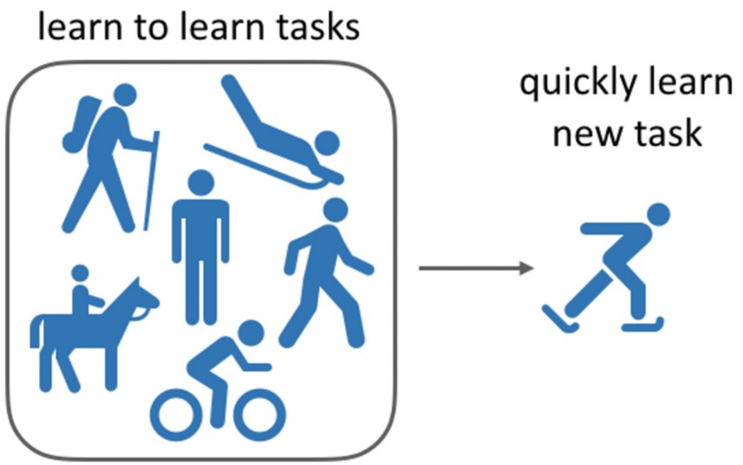


- Few Shot Learning

Recap of Last Lecture

- Multi-task Learning and Meta Learning

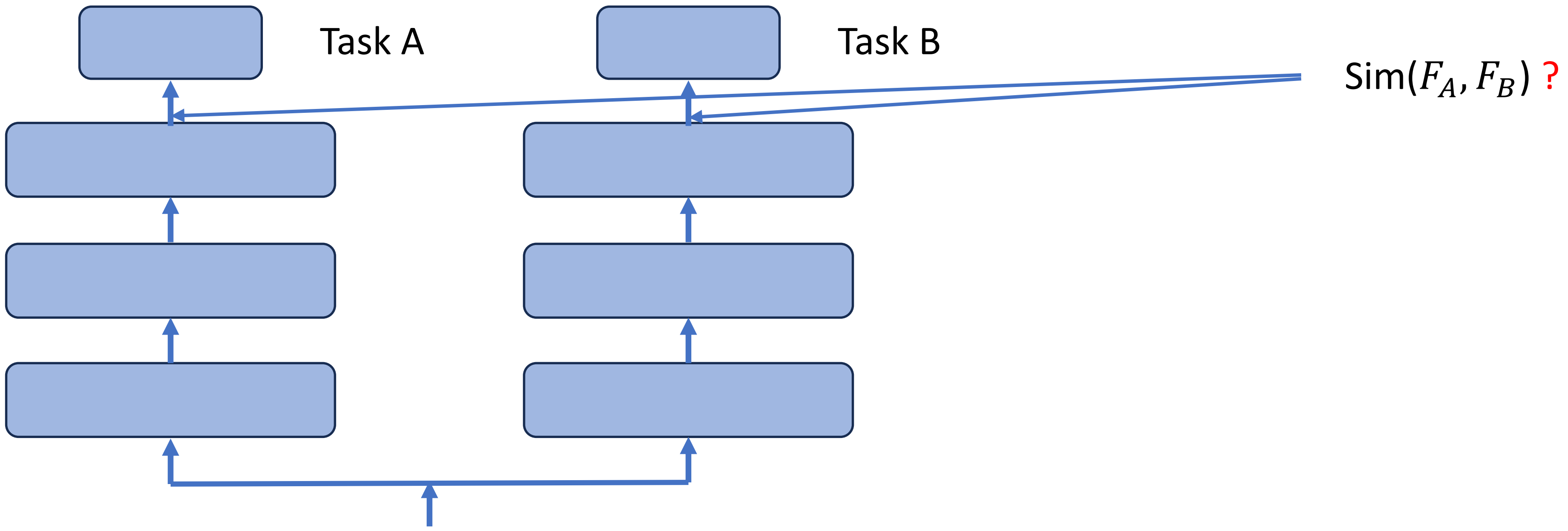




- Works if the tasks are relevant

Task Relevance through Feature Similarity

- Consider one model for each task, with same input



Agenda

- Feature Similarity
- Attribution to Input Feature(s)
- Attribution to Training Sample(s)

Motivate

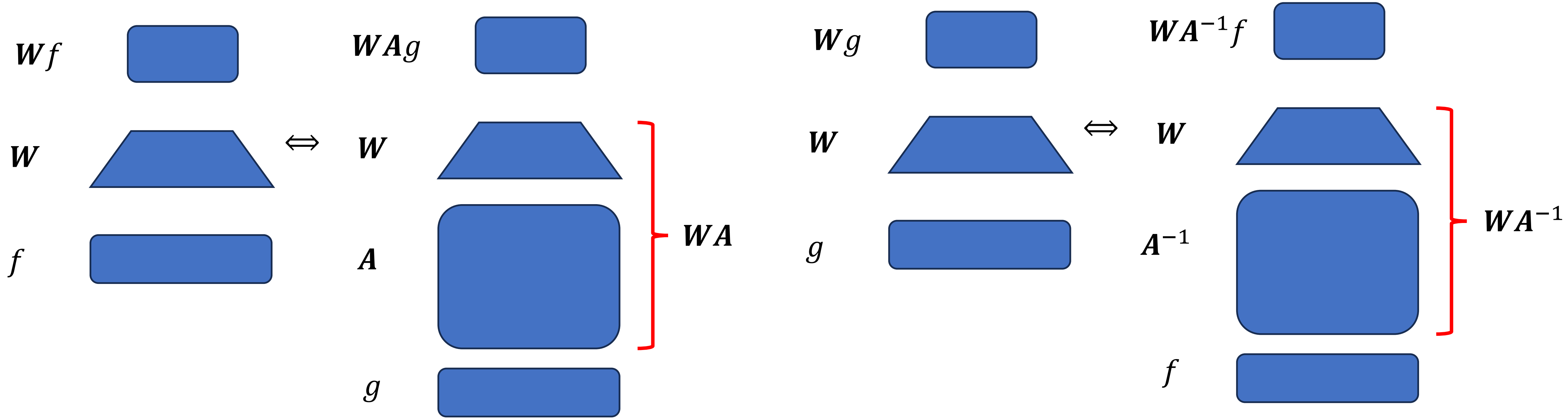
- Compare two sets of representations

$$F = [f_1, \dots, f_n], \qquad G = [g_1, \dots, g_n]$$

  for the same set of inputs $[x_1, \dots, x_n]$
- Many choices
  - $\frac{1}{n}\sum_{i=}^{n}\langle f_i, g_i\rangle$?
  - But what if f_i and g_i are different dimensional?
- Nice if similarity is bounded between 0 and 1
- Also, what if there is some matrix A, such that $F = AG$?

Motivate: linear invariance

- If there is some invertible matrix A, such that $g_i = Af_i$ for any i
- Any classifier built on f is equivalent to another one built on g
- Vice versa



Motivate

- In that sense, features F and G have the same effect
- We want sim(F, G) invariant w.r.t (invertible) linear transform, i.e.,

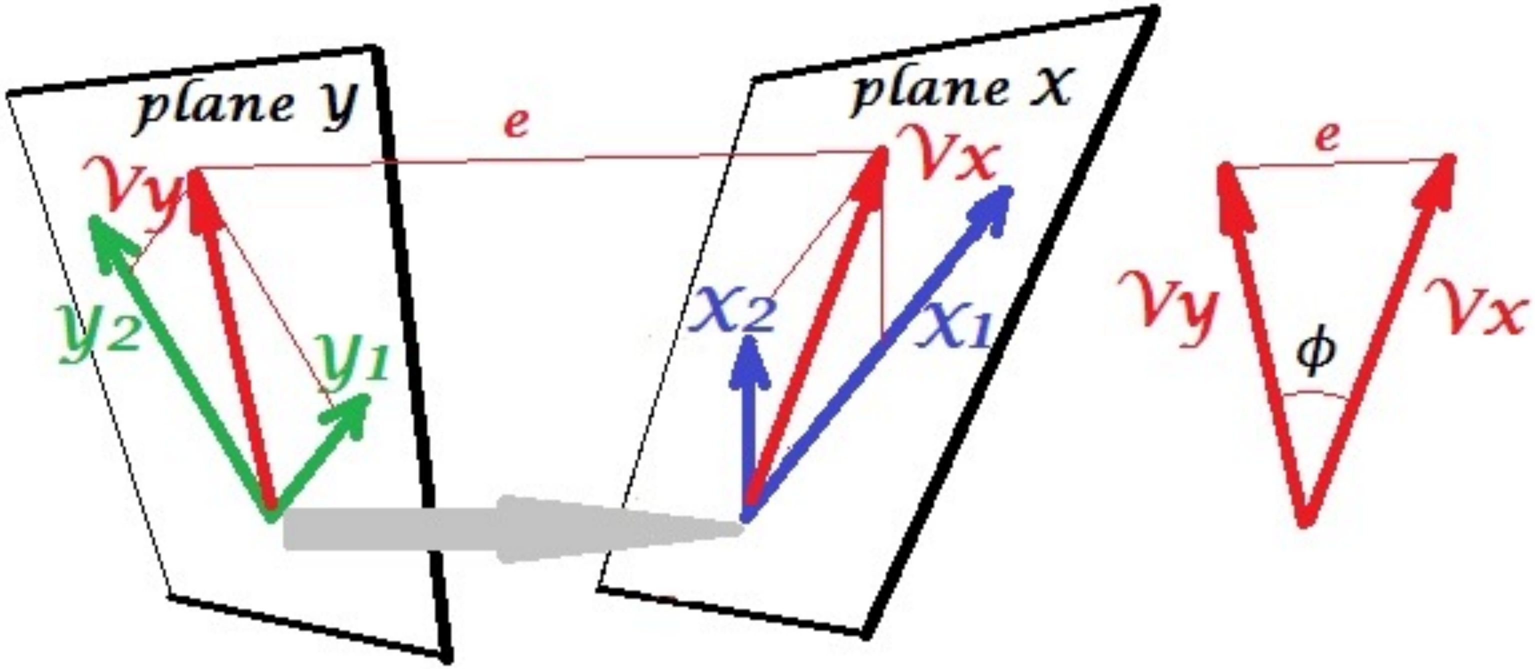
$$\text{sim}(AF, BG) = \text{sim}(F, G)$$

- Summary: Sim(F, G) is preferred to be

1. Bounded between $[0,1]$
2. Invariant w.r.t (invertible) linear transforms

Detour: Canonical Correlation Analysis (CCA)

- Find directions in two sets of features that correlates the most



Cartoon from [stackexchange]()

Formalize: CCA

- Assume F, G are centered (mean subtracted)
- CCA seeks two directions x, y, applied on F and G, such that

$$\max_{x,y}\left\{\rho(x,y) \triangleq \frac{x^T\Sigma_{F,G}\,y}{\sqrt{x^T\Sigma_{F,F}\,x}\sqrt{y^T\Sigma_{G,G}\,y}}\right\}$$

- Massage by leveraging SVD

$$F = U_F\Lambda_F V_F^T \text{ and } G = U_G\Lambda_G V_G^T$$

- Change of variable $\tilde{x} = \Lambda_F U_F^T x$, $\tilde{y} = \Lambda_G U_G^T y$, then

$$\rho(x,y) = \frac{\tilde{x}^T V_F^T V_G \tilde{y}}{\|\tilde{x}\|\|\tilde{y}\|}$$

Solve for CCA

$$\rho(x, y) = \frac{\tilde{x}^T V_F^T V_G \tilde{y}}{\|\tilde{x}\|\|\tilde{y}\|}$$

- Further introduce $\vec{x} = \frac{\tilde{x}}{\|\tilde{x}\|}$, and $\vec{y} = \frac{\tilde{y}}{\|\tilde{y}\|}$, then $\rho(x, y) = \vec{x}^T V_F^T V_G \vec{y}$

So we are seeking unit-norm vectors $\vec{x}$ and $\vec{y}$ Such that

$$\max_{\|\vec{x}\|=1, \|\vec{y}\|=1} \{\rho = \vec{x}^T V_F^T V_G \vec{y}\}$$

- Run SVD of $V_F^T V_G = X \Theta Y^T$
- $\vec{x}^* = X[:, 1]$, $\vec{y}^* = Y[:, 1]$, $\rho^* = \Theta_1$

Nice Property of CCA

- In fact, we can show

$\rho^* \in [0,1]$, and is invariant w.r.t invertible linear transforms on F and G

- ρ^* is a desired a similarity measure!

Recipe:

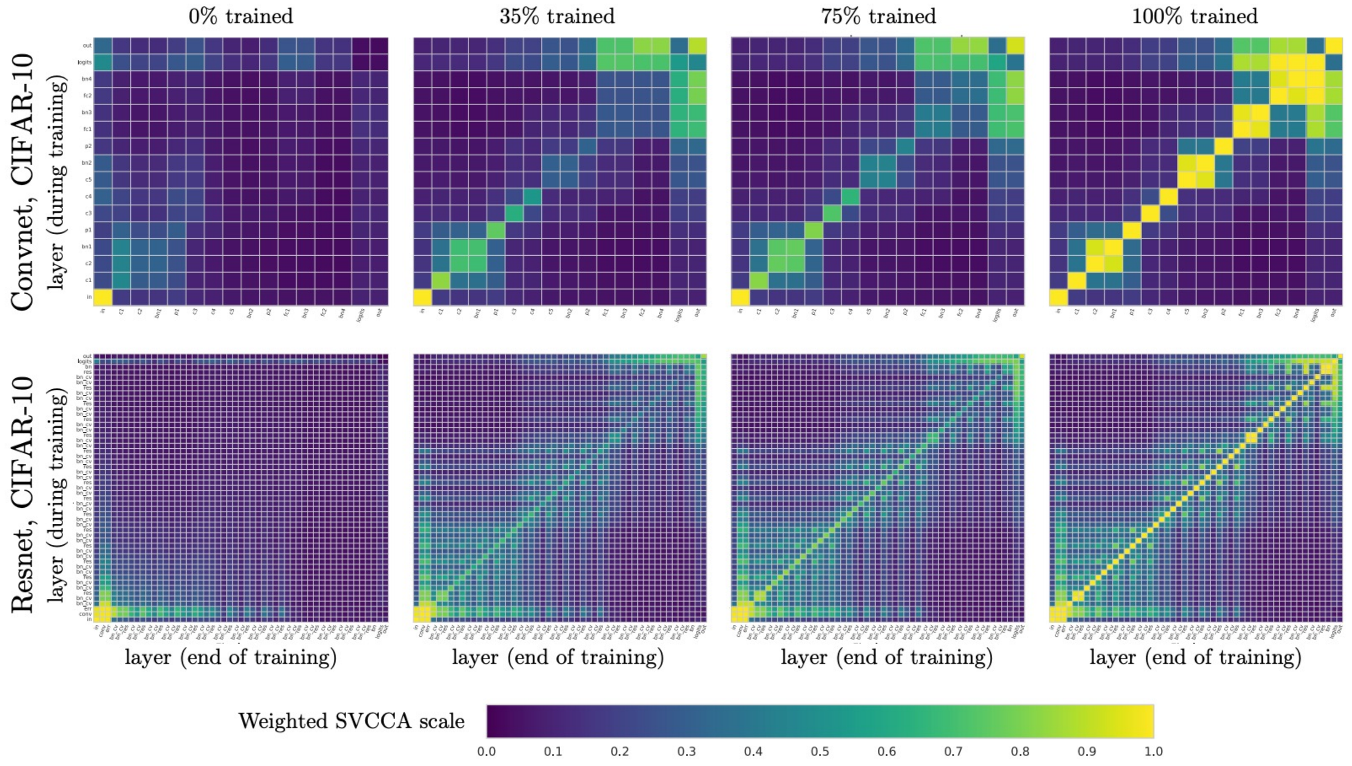
1. Run SVD on F and G

$$F = U_F \Lambda_F V_F^T \text{ and } G = U_G \Lambda_G V_G^T$$

2. Run SVD of $V_F^T V_G$, let the singular values be θ_i
3. Use θ_1 or $\frac{1}{k}\sum_{i=1}^{k} \theta_i^2$ as the similarity index

SVCCA ([Raghu et. al, 2017](#))

- Compare layers at training against after trained
- Bottom layer doesn't change much throughout training



Use SVCCA to Understand training of LMs

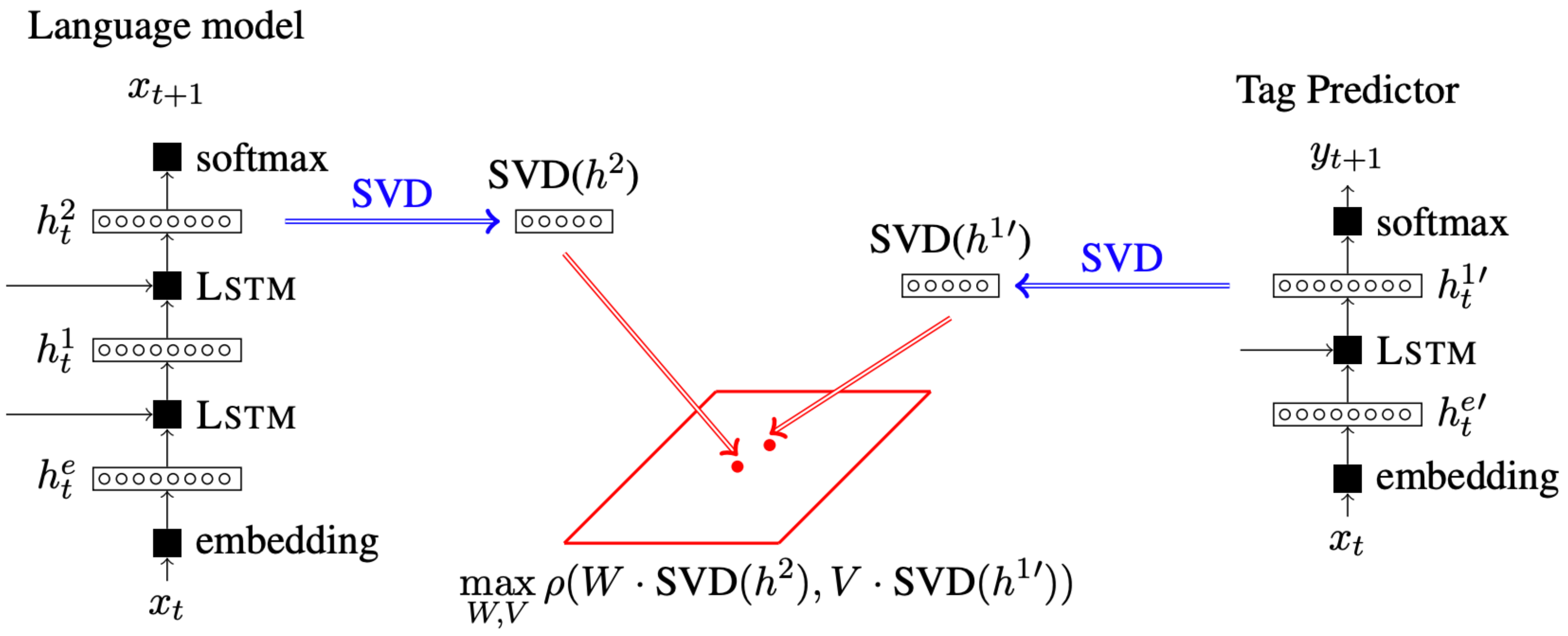
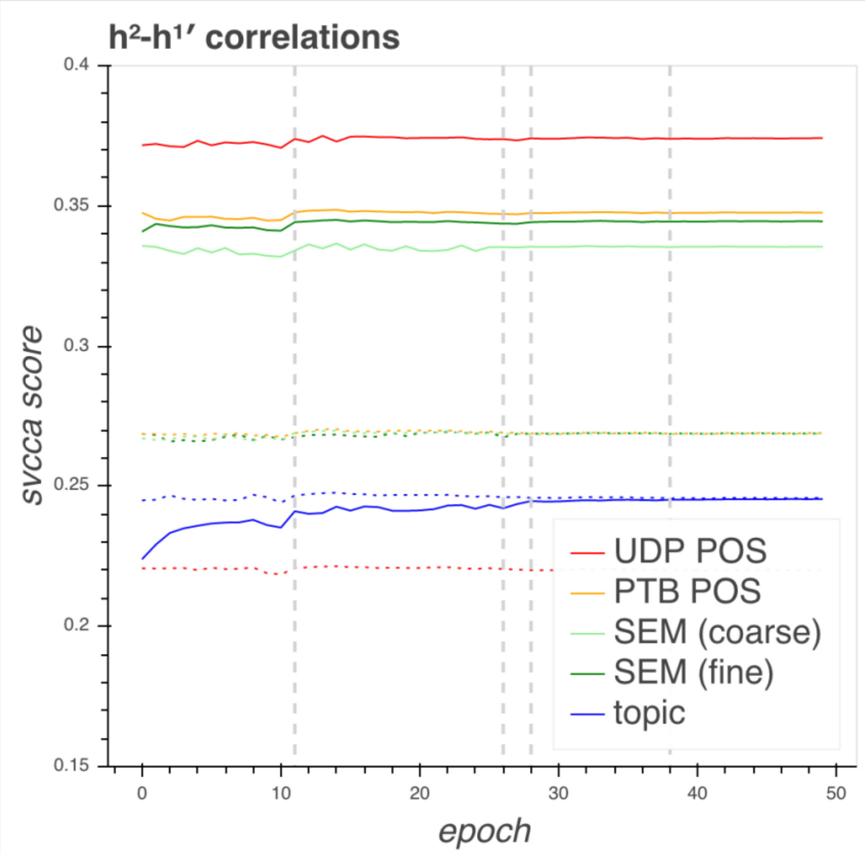


Figure 1: SVCCA used to compare the layer h^2 of a language model and layer $h^{1\prime}$ of a tagger.

[Saphra. et. al, 2019]

Use SVCCA to Understand training of LMs

- Three taggers:
  - POS (part of speech), e.g., noun vs. verb
  - Semantic, e.g., cash (noun) vs. cash (verb)
  - topic
- LM feature:
  - more similar to feature for POS tagger
  - not so much to feature for topic tagger
- Why so?



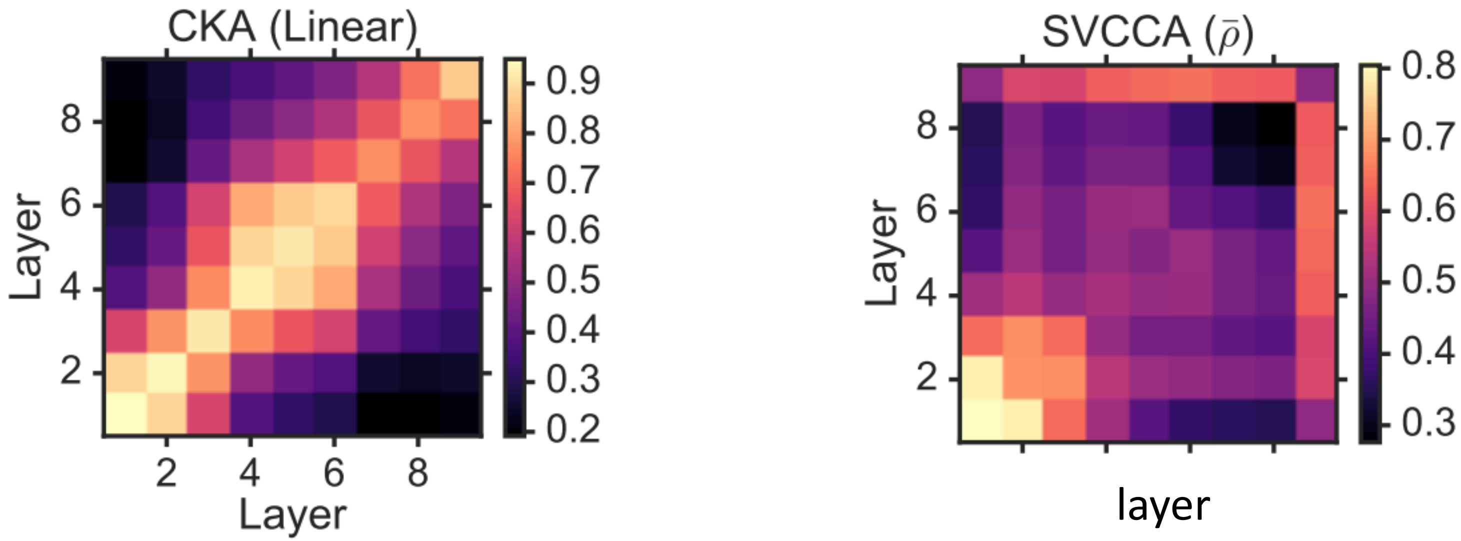
Another similarity: CKA (Kornblith et. al, 2019)

- We may also ask sim(AF, BG)=sim(F, G) for any rotation matrix
$$A, B \text{ such that } AA^T = I \text{ and } BB^T = I$$
- Build inter-sample similarity (kernel): assume centered features
$$\kappa_F(i,j) = f_i^T f_j,\ \kappa_G(i,j) = g_i^T g_j$$
- Compute Centered Kernel Alignment (CKA)
$$r = \frac{\langle \kappa_F, \kappa_G \rangle}{\sqrt{\langle \kappa_F, \kappa_F \rangle} \cdot \sqrt{\langle \kappa_G, \kappa_G \rangle}}$$

$r \in [0,1]$, and is invariant w.r.t rotation on F and G

CCA vs CKA

- CKA is more “correct”
- Compare two models with the same architecture, trained from different initializations



- Corresponding layers should be more similar, but not captured by CCA

[Kornblith et. al, 2019](#)

CKA: More Observations

- Layer similarity implies when the model starts to overfit as it grows deeper

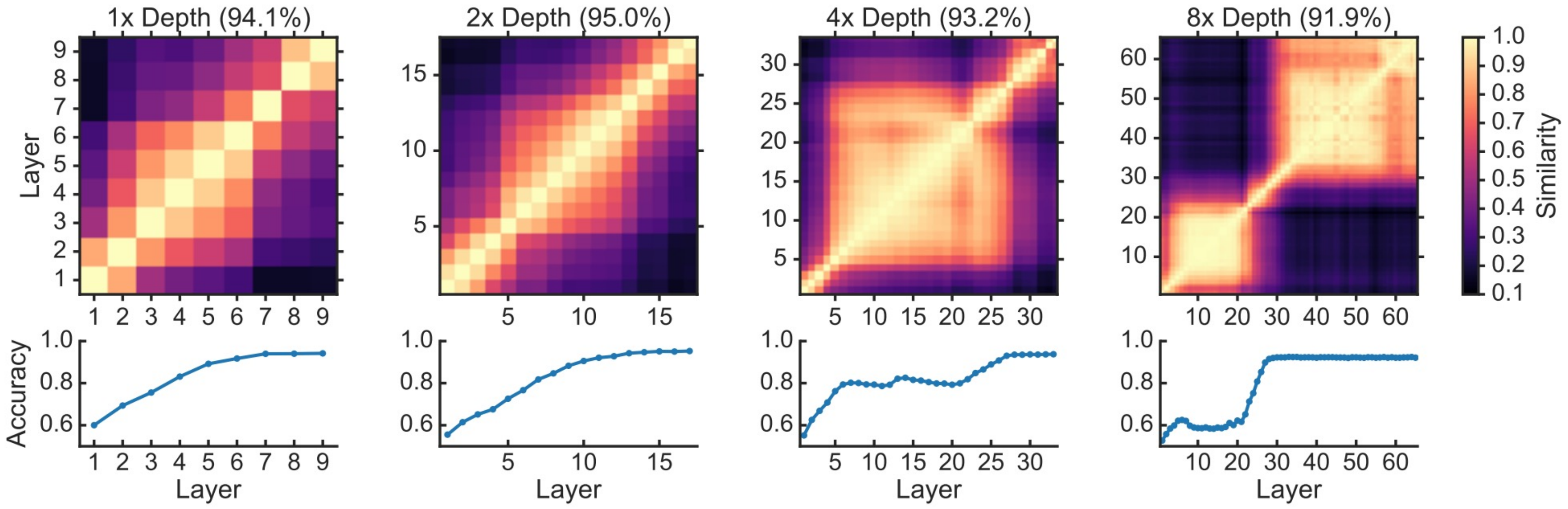


Figure 3. CKA reveals when depth becomes pathological. **Top**: Linear CKA between layers of individual networks of different depths on the CIFAR-10 test set. Titles show accuracy of each network. Later layers of the 8x depth network are similar to the last layer. **Bottom**: Accuracy of a logistic regression classifier trained on layers of the same networks is consistent with CKA.

[Kornblith et. al, 2019](#)

Using CKA to Understand Transfer Learning

- Transfer from Pretrained Network on ImageNet to X-ray images
- Transfer Learned networks are more similar than those learned from scratch

Table 1: Feature similarity for different layers of ResNet-50, target domain CHEXPERT

models/layer	conv1	layer 1	layer 2	layer 3	layer 4
P-T & P	0.6225	0.4592	0.2896	0.1877	0.0453
P-T & P-T	0.6710	0.8230	0.6052	0.4089	0.1628
P-T & RI-T	0.0036	0.0011	0.0022	0.0003	0.0808
RI-T & RI-T	0.0016	0.0088	0.0004	0.0004	0.0424

P: pretrained on Imagenet; P-T: transferred to X-ray images; RI-T: learned from scratch on X-ray images

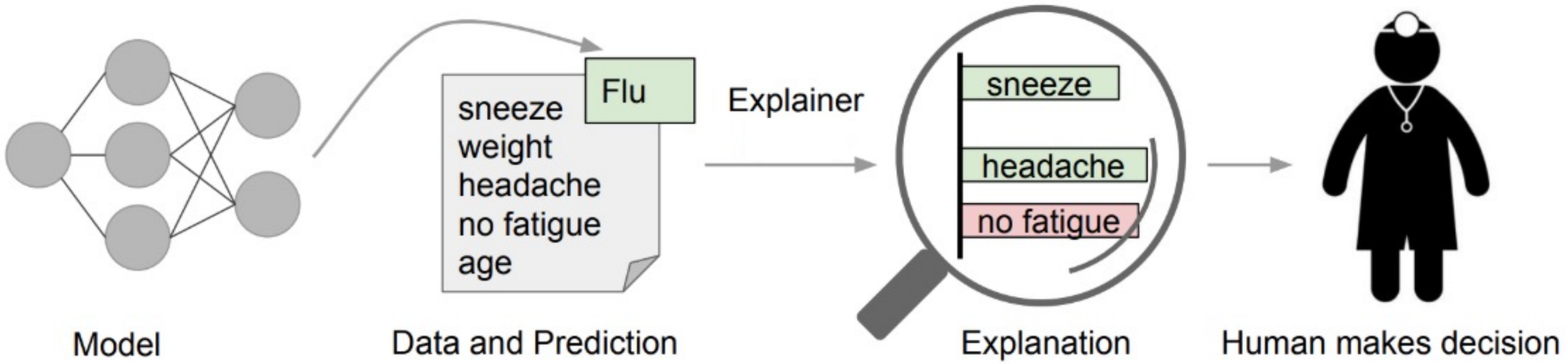
[Neyshabur et. al, 2020]

Agenda

- Feature Similarity
- Attribution to Input Feature(s)
- Attribution to Training Sample(s)

Motivating: Interpreting Deep Models

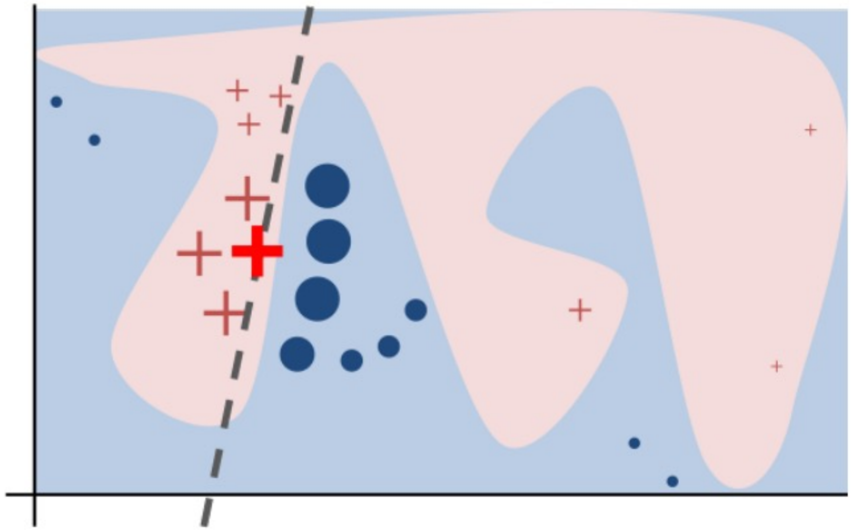
- How can we trust deep model's decision?



(Ribeiro et. al, 2016): Explainer (another model) provides insights that help human to understand deep models

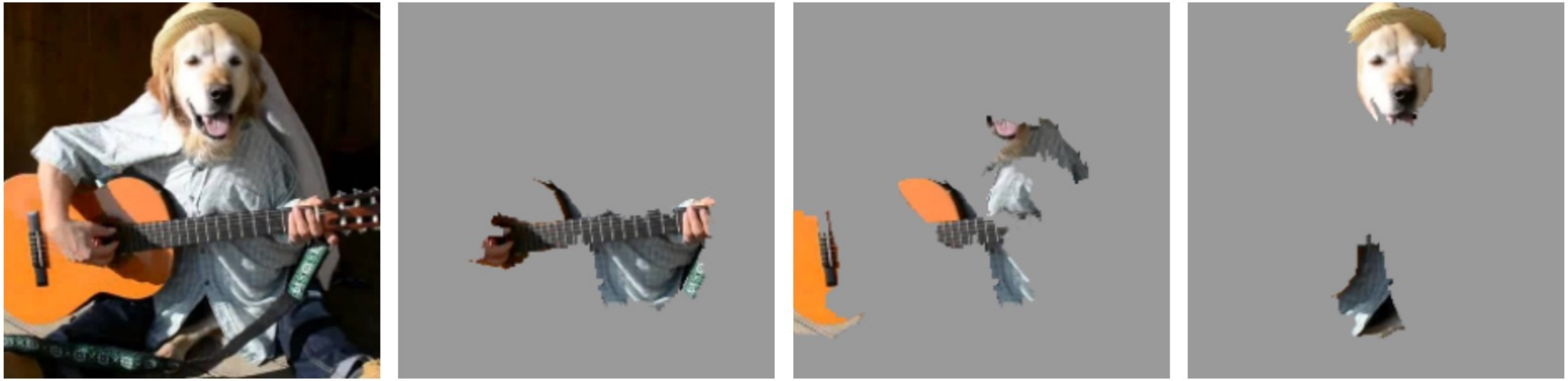
Motivating: Explainer as Linear Model

- Linear Models are more Interpretable
- Consider $\hat{y} = \sum_{i=1}^{d} \alpha_i x_i$
- $|\alpha_i|$ indicates the importance of i-th feature
- Especially when d is small, or many $\alpha_i = 0$
- Approximate deep network with (locally) sparse linear model?



Ribeiro et. al, 2016

What we want to Achieve



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

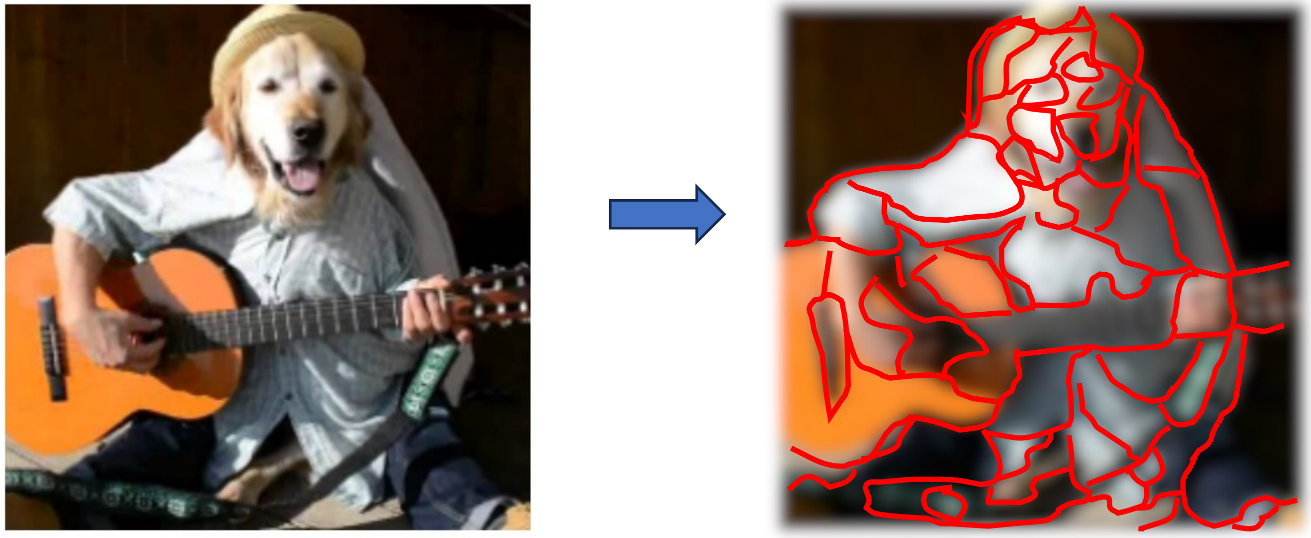
Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$)**, "Acoustic guitar"** ($p = 0.24$) **and "Labrador"** ($p = 0.21$)

[Ribeiro et. al, 2016]

Local Interpretable Model-agnostic Explanations

(abbr. **LIME**) To explain an input $\boldsymbol{x}$, with decision $f(\boldsymbol{x})$ made by network

- Convert to its "interpretable" version $\boldsymbol{x}'$, e.g., super-pixel segmented



Q: Why not just work on the original image?

- Fit a linear model around $\boldsymbol{x}'$

Ribeiro et. al, 2016

LIME

- Create multiple perturbations around $\boldsymbol{x}'$
  - e.g., $+\varepsilon$ for a super-pixel, denoted as $\boldsymbol{z}'$
  - Input $\boldsymbol{z}'$ to the network to get decision $f(\boldsymbol{z}')$
- On the dataset $\{\boldsymbol{z}'_n, f(\boldsymbol{z}'_n)\}$ train a sparse linear model

$$\min_{\|\boldsymbol{\alpha}\|_0 \leq K} \frac{1}{N} \sum_{n=1}^{N} \|\langle \boldsymbol{\alpha}, \boldsymbol{z}'_n \rangle - f(\boldsymbol{z}'_n)\|^2$$

- Many solvers: e.g., [sklearn.linear_model.lasso]

[Ribeiro et. al, 2016]

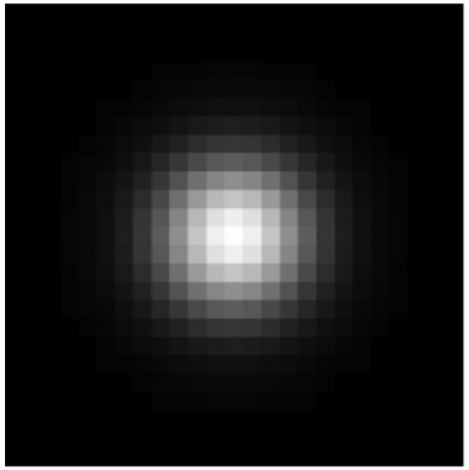
Weighted Loss

- Down-weigh the loss if ε is very big

$$\min_{\|\boldsymbol{\alpha}\|_0 \leq K} \frac{1}{N} \sum_{n=1}^{N} w(\boldsymbol{z}_n', \boldsymbol{x}) \|\langle \boldsymbol{\alpha}, \boldsymbol{z}_n' \rangle - f(\boldsymbol{z}_n')\|^2$$

- RBF kernel

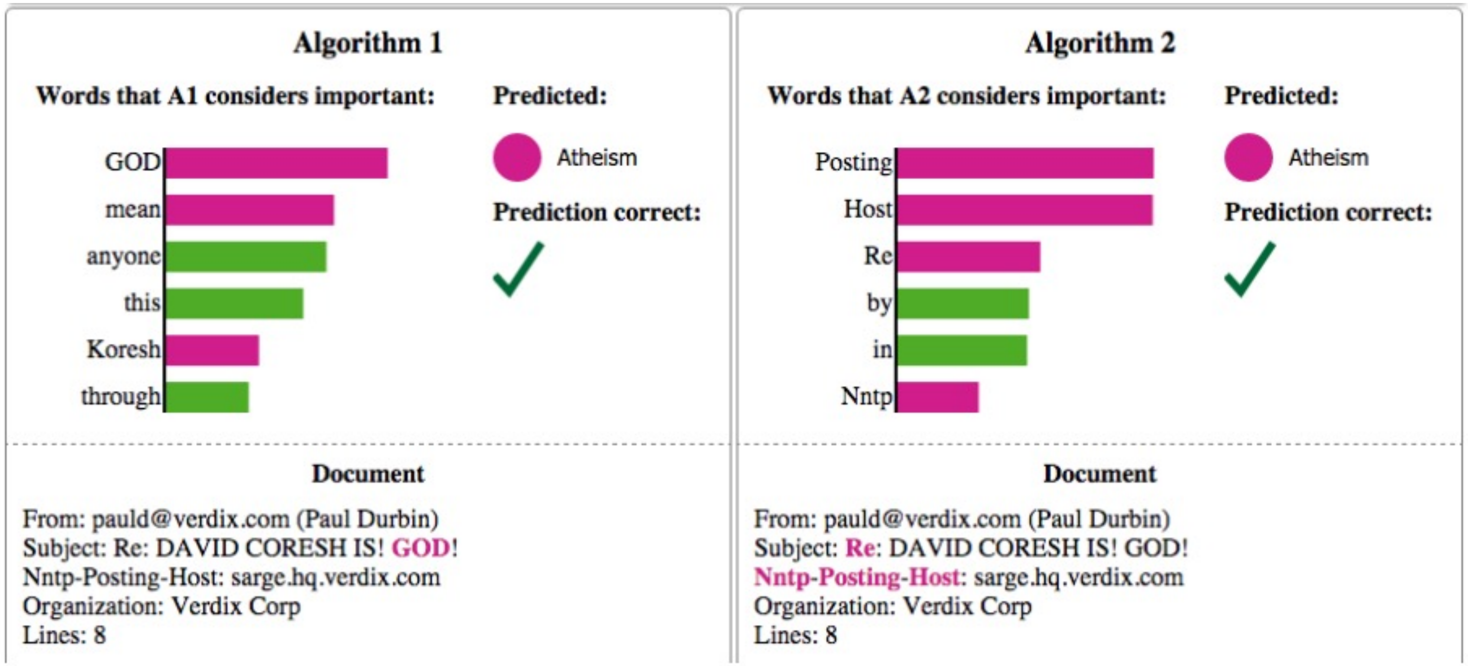
$$w(\boldsymbol{z}_n', \boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{z}_n' - \boldsymbol{x}\|^2}{\sigma^2}\right)$$



- Choice of σ^2 is not clear

Discussion

- How do we apply LIME for a text classification model?



Explain the distinct decisions of two text classifiers

- Why not just calculate the gradient w.r.t $\boldsymbol{x}$?

[Ribeiro et. al, 2016](#)

Saliency map

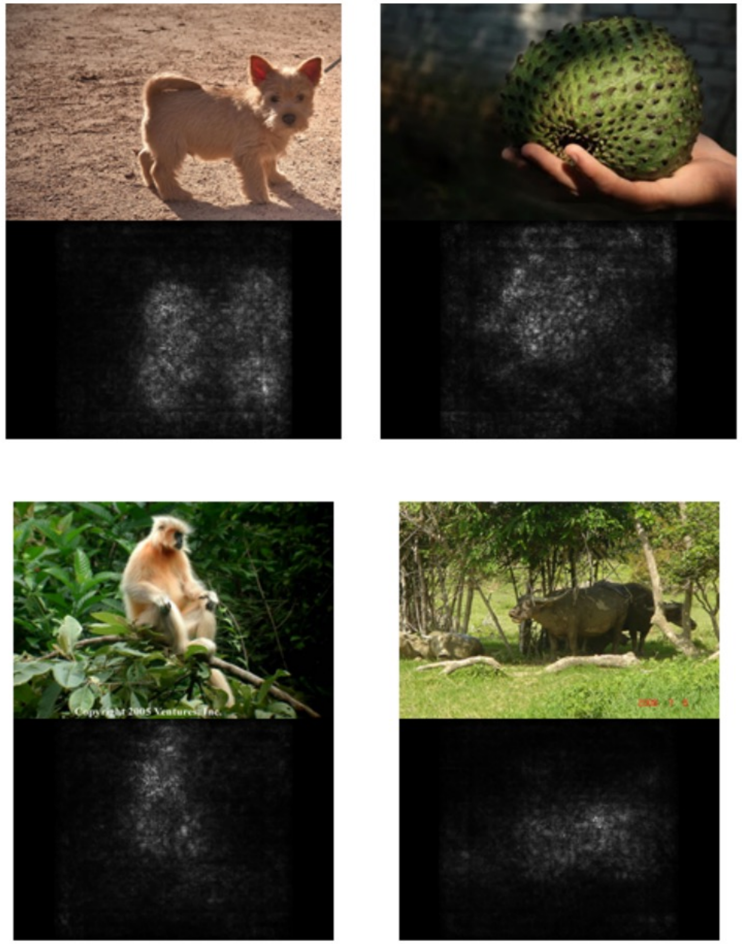
- Yes! There are earlier intuitions in Computer Vision
- Consider 1st order Taylor Expansion

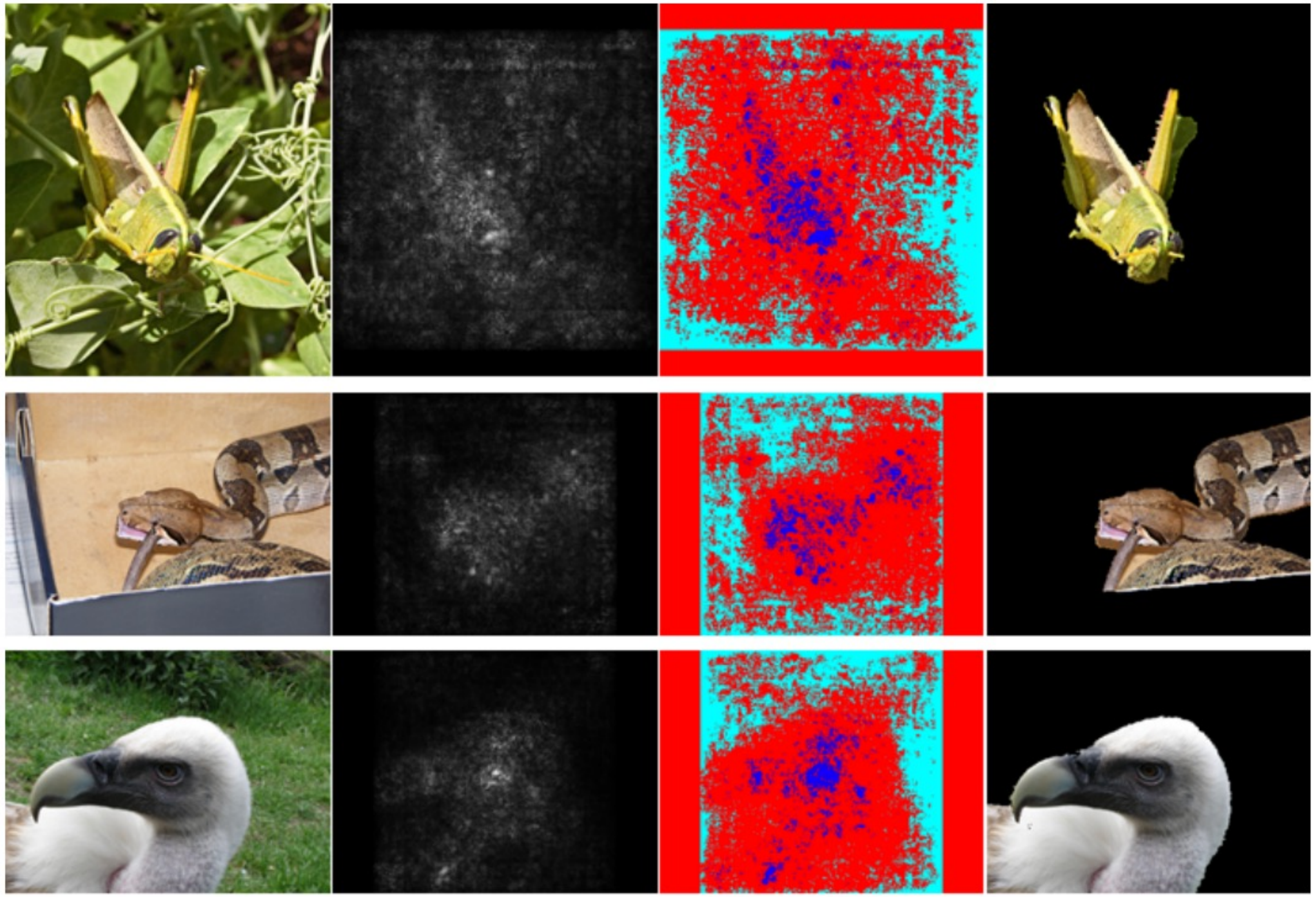
$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + \left.\frac{\partial f}{\partial x}\right|_{\boldsymbol{x}=\boldsymbol{x}_0} (\boldsymbol{x} - \boldsymbol{x}_0)$$

- e.g. $\boldsymbol{x}$ as input image
- $f(\boldsymbol{x})$ as network predicted probability of class label
- Visualize $\left.\frac{\partial f}{\partial x}\right|_{\boldsymbol{x}=\boldsymbol{x}_0}$

[Simonyan et. al, 2013]

Saliency map Examples



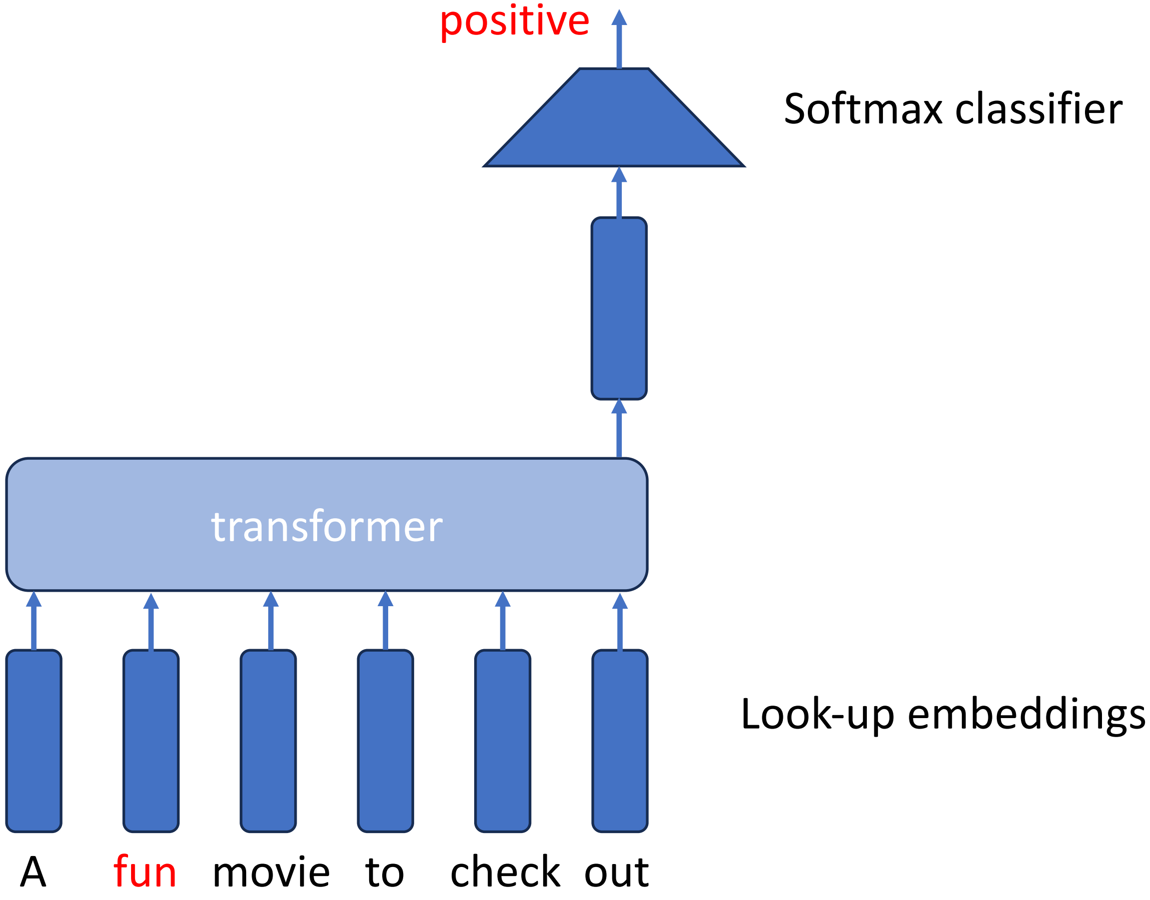


Object localization using the saliency map. Note: No bounding box training data used

[Simonyan et. al, 2013]

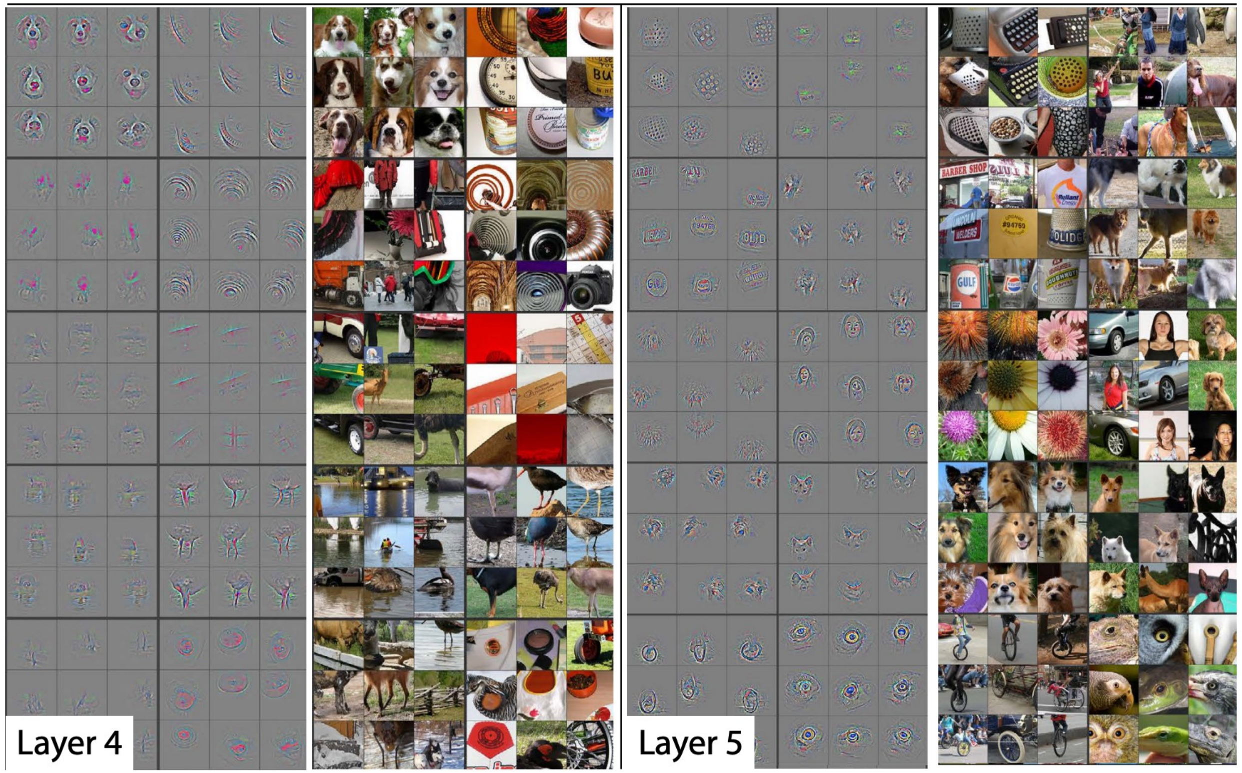
Discussion

- How to obtain saliency map in a text classifier?

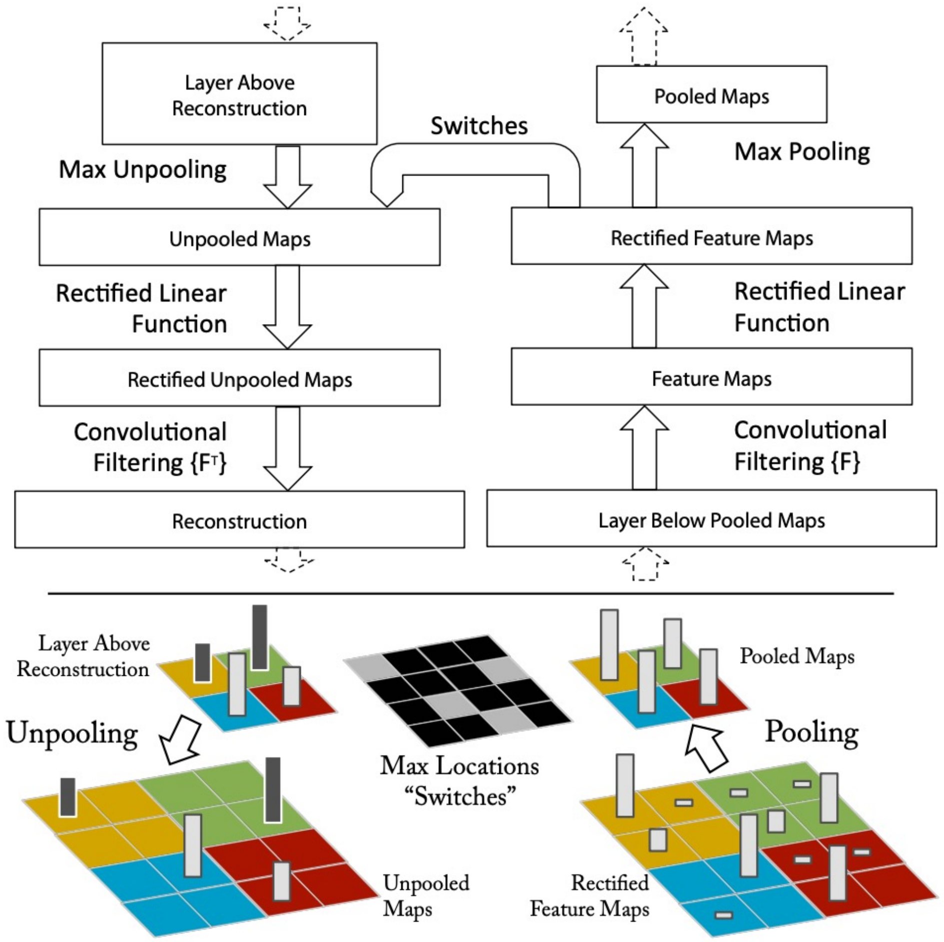


Revisit Visualization of Feature Maps in Lecture 3

- DeconvNet (Zeiler & Fergus, 2013)

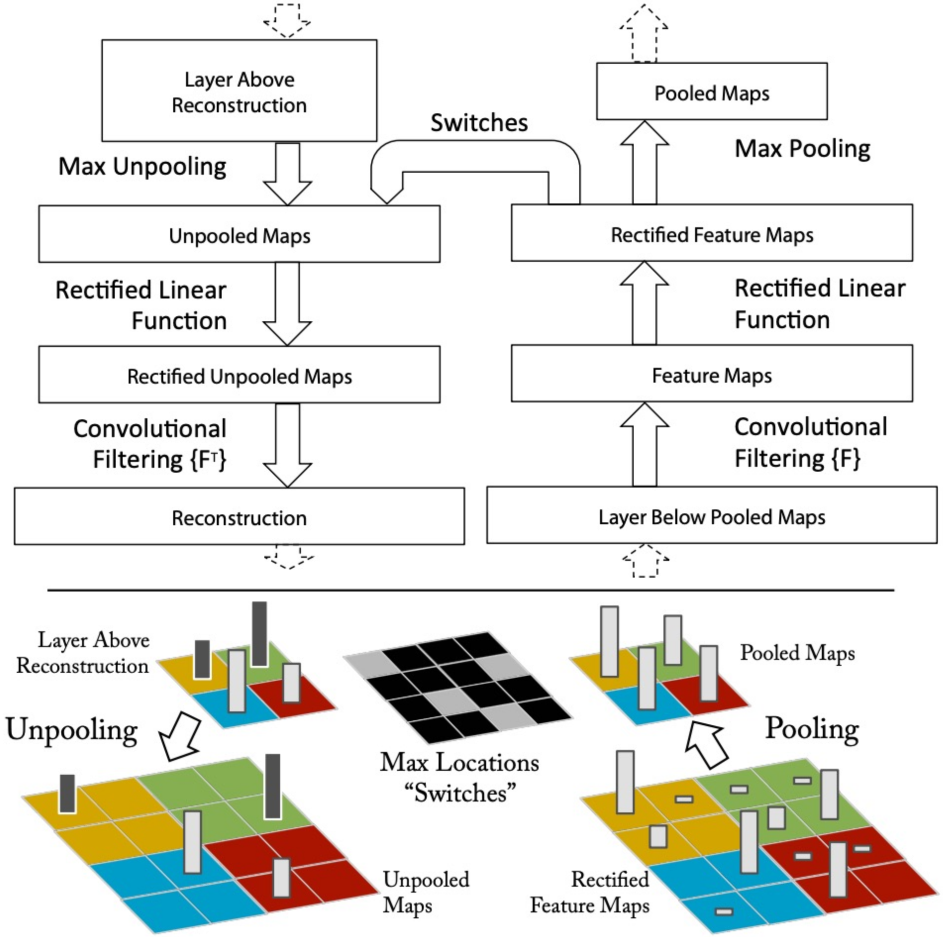


DeconvNet ([Zeiler & Fergus, 2013](#))



- Feature maps are smaller than input image
- "Project" Feature map back to pixel space
- Max "unpooling":
  - put back the max value to where it sat
  - put 0 for other locations in the region
- Transposed convolution: $\star\, \kappa^T$

DeconvNet Architecture



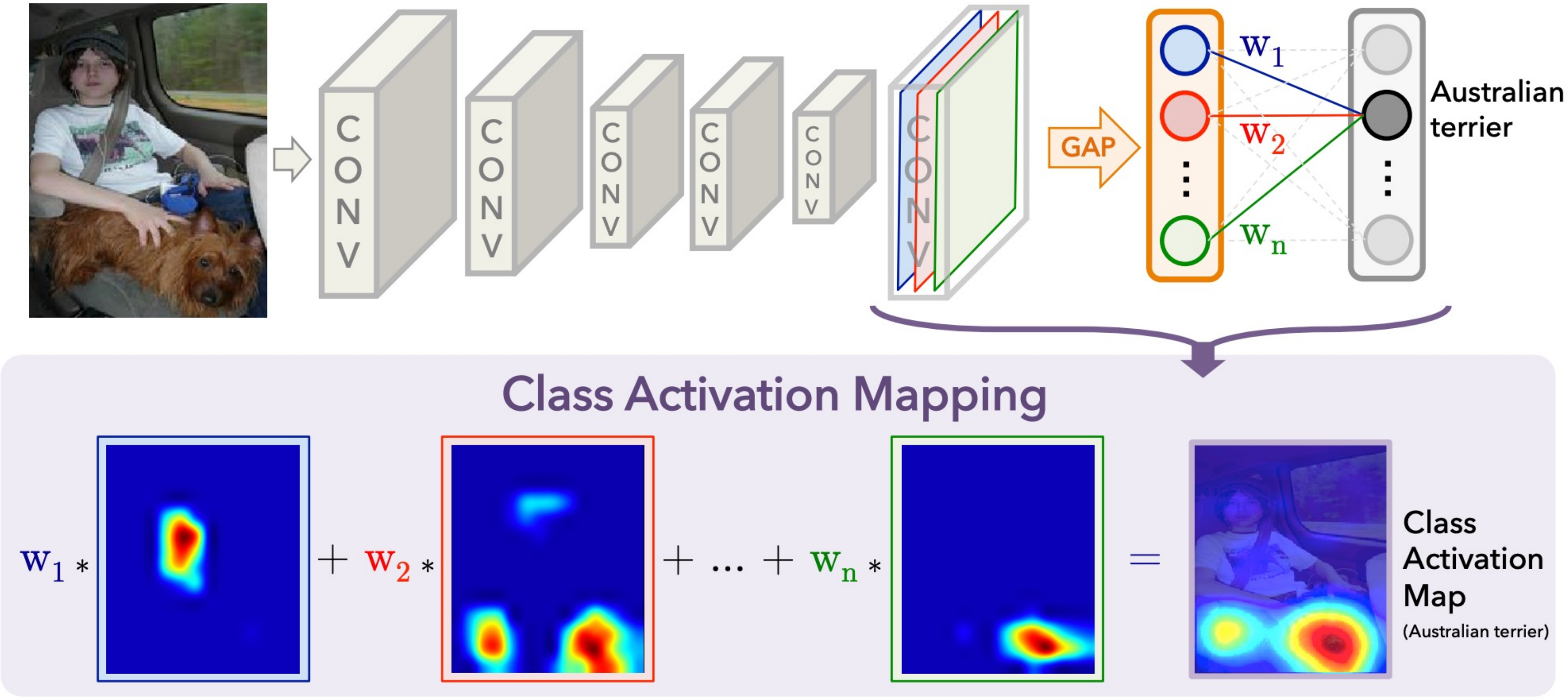
- Gradient of pooling layer:
  - Place $\frac{\partial}{\partial O}$ to where the output sat
  - Place 0 to other locations in the region
- Convolute κ^T : recall lecture 3

$$\frac{\partial}{\partial I} = \frac{\partial}{\partial O} \star \kappa^T$$

- DeconvNet computes gradient to some extent!

Class Activation Mapping (CAM)

- Insert a Global Average Pooling (GAP) Layer before softmax classifier

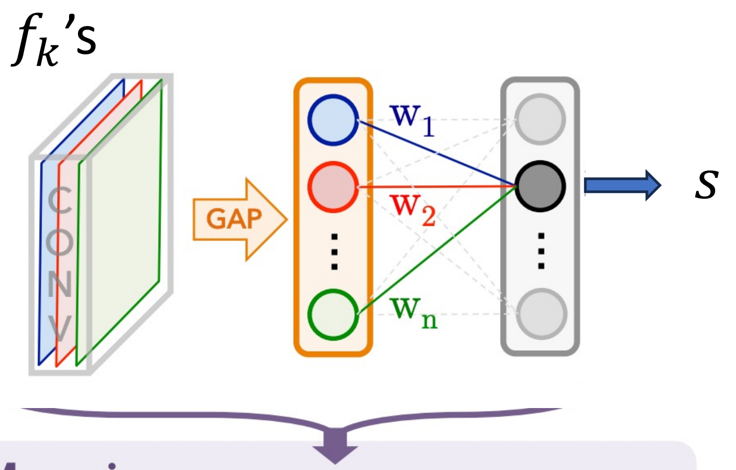


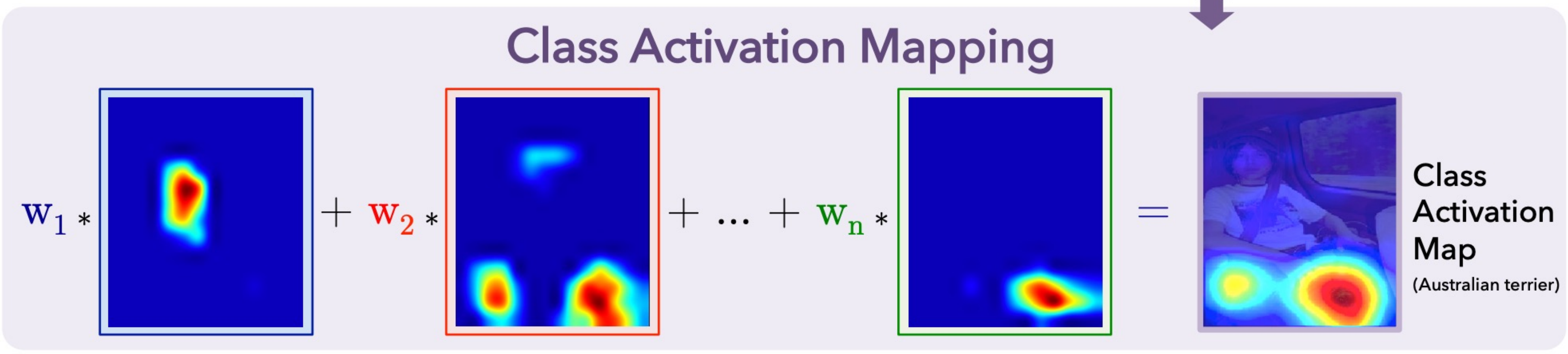
[Zhou et. al, 2015]

How GAP works

- $f_k[x,y]$: k-th feature map
- Global Average Pooling: $\sum_{x,y} f_k[x,y]$
- Class score

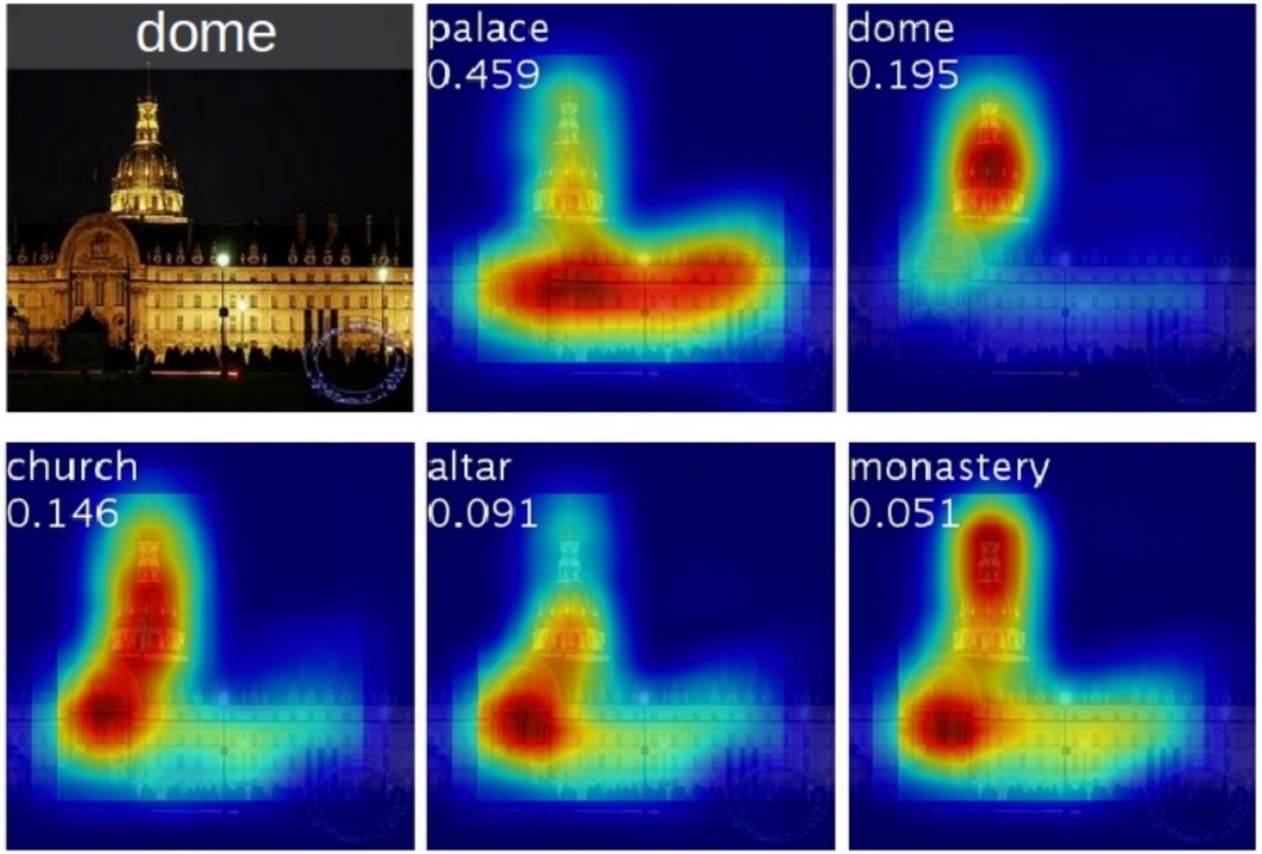
$$s = \sum_k w_k \sum_{x,y} f_k[x,y] = \sum_{x,y} \boxed{\sum_k w_k f_k[x,y]}$$





How CAM works

- Upsample the Class Activation Map to Image size



Trained for classification only but can achieve localization!

Grad-CAM

- Weighted average of feature maps at middle layer

$$w_k f_k[x, y]$$

where

$$w_k = \sum_{x,y} \frac{\partial s}{\partial f_k[x, y]}$$

- Discussion:
  - Show that CAM is a special case of Grad-CAM
  - Generalize to text input?

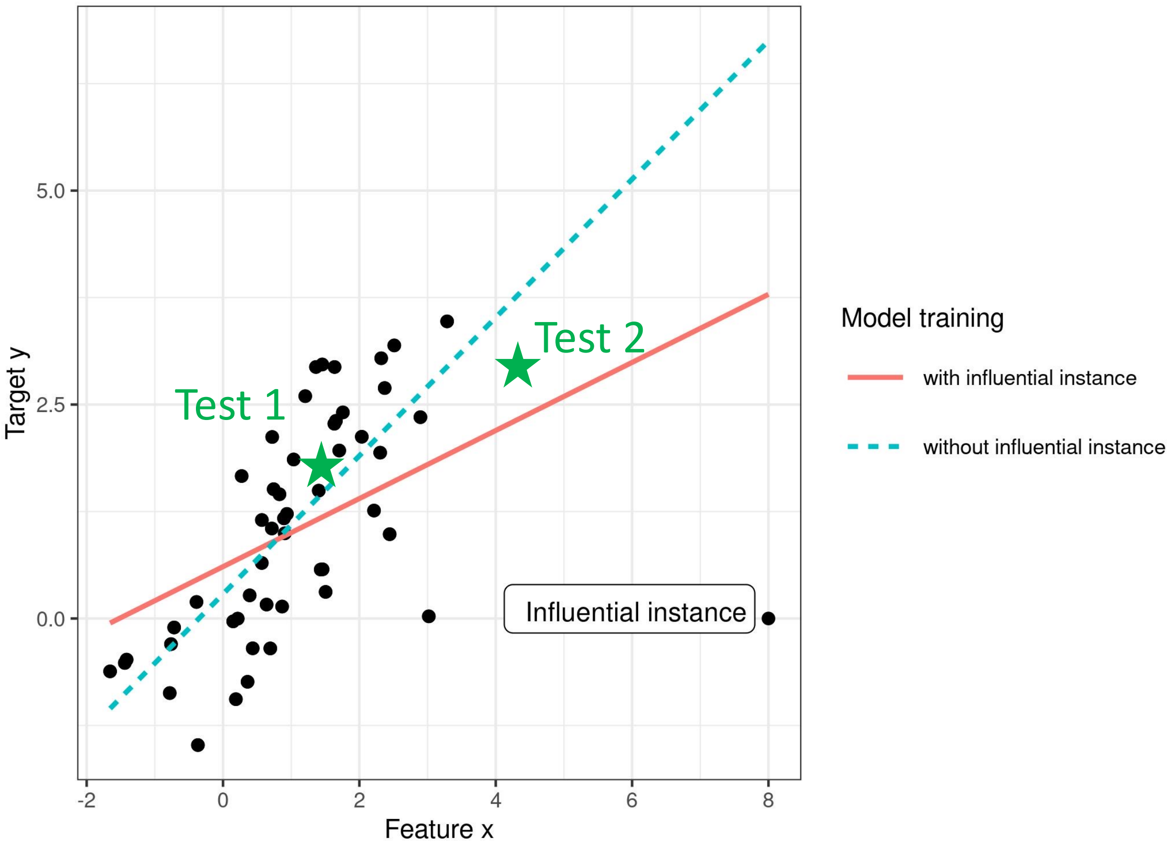
Selvaraju et. al, 2019

Agenda

- Feature Similarity
- Attribution to Input Feature(s)
- Attribution to Training Sample(s)

Motivation

- Influential training instances



- How to identify them?
- Influence to a test sample?

  Test 1 vs Test 2

Influence Function

- Introduced in the 1970s in the field of robust statistics
- Consider an estimator T that acts on a distribution p
- How much does T change if we perturb p

Koh & Liang, 2020, slide adapted from ece739 lecture slides

Formalize

- Trained model parameter

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; z_i)$$

- Perturb a training sample z by additionally weighing ε on its loss

$$\hat{\theta}_{\varepsilon} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; z_i) + \varepsilon \ell(\theta; z)$$

- e.g., removing the sample amounts to $\varepsilon = -\frac{1}{n}$
- By construction, $\hat{\theta} \equiv \hat{\theta}_0$
- The influence on a test sample $\mathcal{z}$ is

$$\ell\left(\hat{\theta}_{\varepsilon}; \mathcal{z}\right) - \ell\left(\hat{\theta}_0; \mathcal{z}\right)$$

Derivations

$$\hat{\theta}_\varepsilon = \arg\min_\theta \frac{1}{n}\sum_{i=1}^{n} \ell(\theta; z_i) + \varepsilon\ell(\theta; z)$$

- So $\hat{\theta}_\varepsilon$ satisfies

$$\nabla_\theta \frac{1}{n}\sum_{i=1}^{n} \ell(\hat{\theta}_\varepsilon; z_i) + \varepsilon \cdot \nabla_\theta \ell\left(\hat{\theta}_\varepsilon; z\right) = 0$$

- Take derivative w.r.t. ε, and make $\varepsilon \to 0$

$$\underbrace{\left[\nabla_\theta \nabla_\theta \frac{1}{n}\sum_{i=1}^{n} \ell(\hat{\theta}_\varepsilon; z_i)\right]}_{\text{Hessian} \succ \mathbf{0}} \frac{d\hat{\theta}_\varepsilon}{d\varepsilon} + \nabla_\theta \ell\left(\hat{\theta}_\varepsilon; z\right) + \varepsilon\left[\nabla_\theta \nabla_\theta \ell\left(\hat{\theta}_\varepsilon; z\right)\right] \frac{d\hat{\theta}_\varepsilon}{d\varepsilon} = 0$$

$\hat{\theta}_0$ 0 $\hat{\theta}_0$

Derivations

$$\frac{d\hat{\theta}_\varepsilon}{d\varepsilon} = -\underbrace{\left[\nabla_\theta \otimes \nabla_\theta \frac{1}{n}\sum_{i=1}^{n} \ell(\hat{\theta}_0; z_i)\right]^{-1}}_{\boldsymbol{H}^{-1}} \cdot \nabla_\theta\, \ell(\hat{\theta}_0; z)$$

- Influence on test sample z:

$$\ell(\hat{\theta}_\varepsilon; z) - \ell(\hat{\theta}_0; z) \approx \langle \nabla_\theta \ell(\hat{\theta}_0; z), \hat{\theta}_\varepsilon - \hat{\theta}_0 \rangle$$

$$\approx -\varepsilon \cdot \nabla_\theta \ell(\hat{\theta}_0; z)^T \boldsymbol{H}^{-1} \nabla_\theta \ell(\hat{\theta}_0; z)$$

Practical Meaning

- Influence on test sample z:

$$\ell(\hat{\theta}_\varepsilon; z) - \ell(\hat{\theta}_0; z) \approx -\varepsilon \cdot \nabla_\theta \ell(\hat{\theta}_0; z)^T \boldsymbol{H}^{-1} \nabla_\theta \ell(\hat{\theta}_0; z)$$

- $\ell(\hat{\theta}_\varepsilon; z) - \ell(\hat{\theta}_0; z) > 0$, harmful perturbation
- $\ell(\hat{\theta}_\varepsilon; z) - \ell(\hat{\theta}_0; z) < 0$, helpful perturbation
- Removing training sample $z \Leftrightarrow \varepsilon = -\frac{1}{n}$,

  If training set small (small n), big impact
- Ignore Hessian ($\boldsymbol{H}^{-1} \to \boldsymbol{I}$), z close to z, big impact

Cartoon from CS239 slides

Practice

- Derive the Impact for Logistic Regression
- $z = (\boldsymbol{x}, y)$, and $p(y|x) = \sigma(y\theta^T \boldsymbol{x})$
- Influence on $z = (\boldsymbol{x}_{test}, y_{test})$:

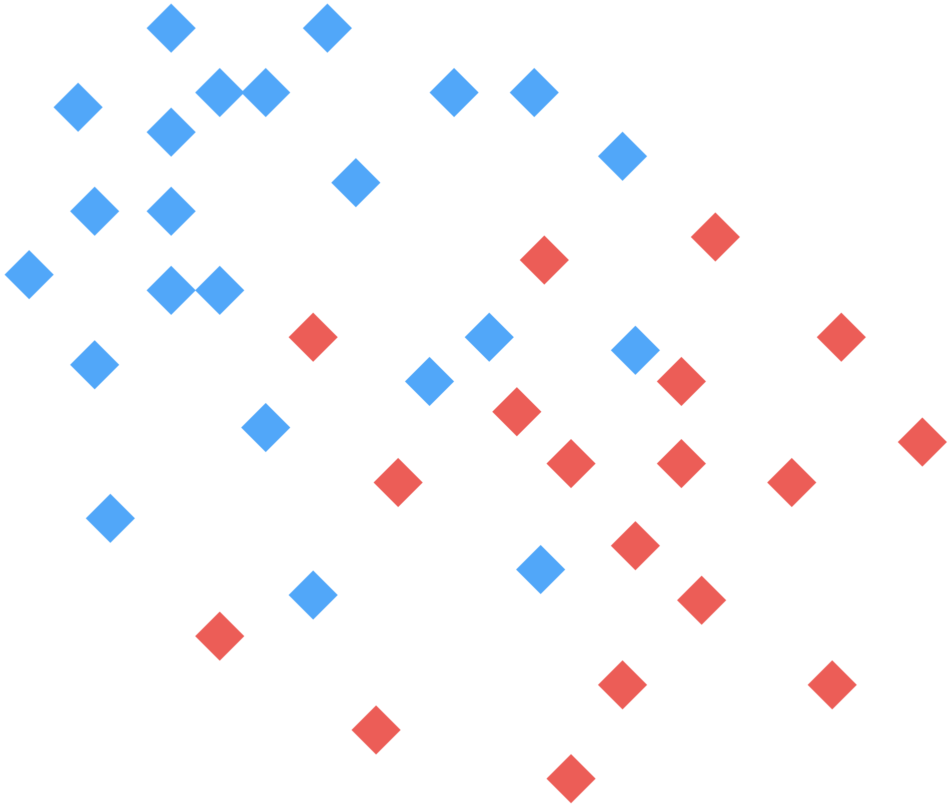
$$- \sigma(-y_{test}\theta^T \boldsymbol{x}_{test}) \cdot \sigma(-y\theta^T \boldsymbol{x}) \cdot \boldsymbol{x}_{test}^T H_\theta^{-1} \boldsymbol{x} \cdot y_{test} y$$

Influence due to closeness of labels

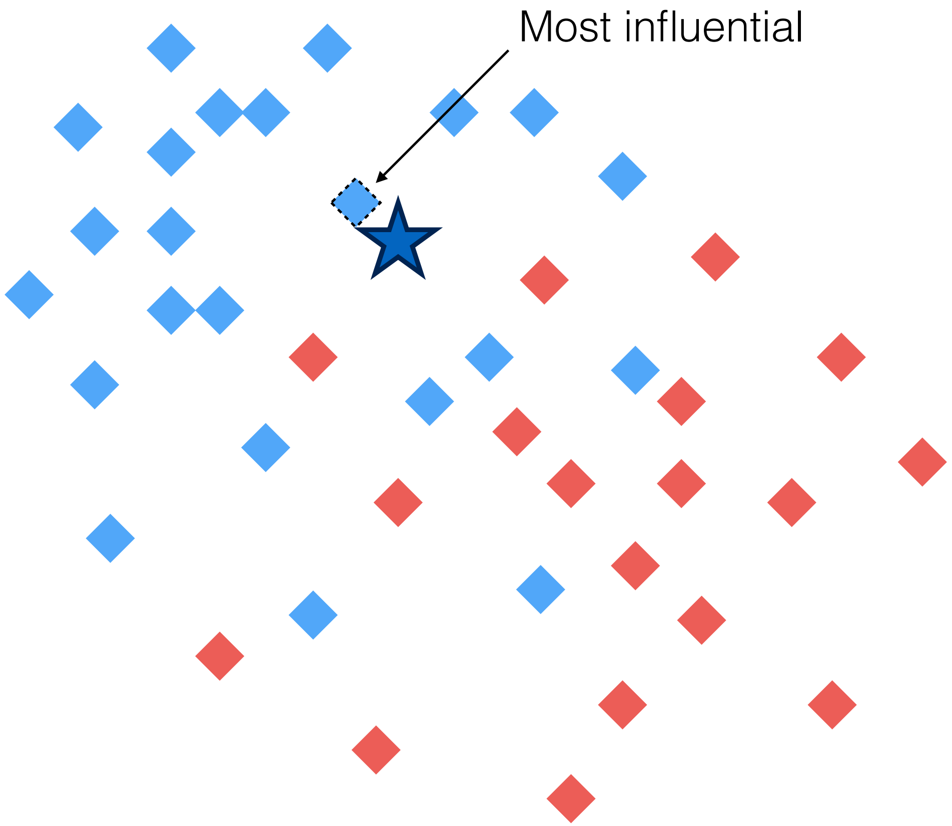
Influence of training loss

Influence due to closeness of features

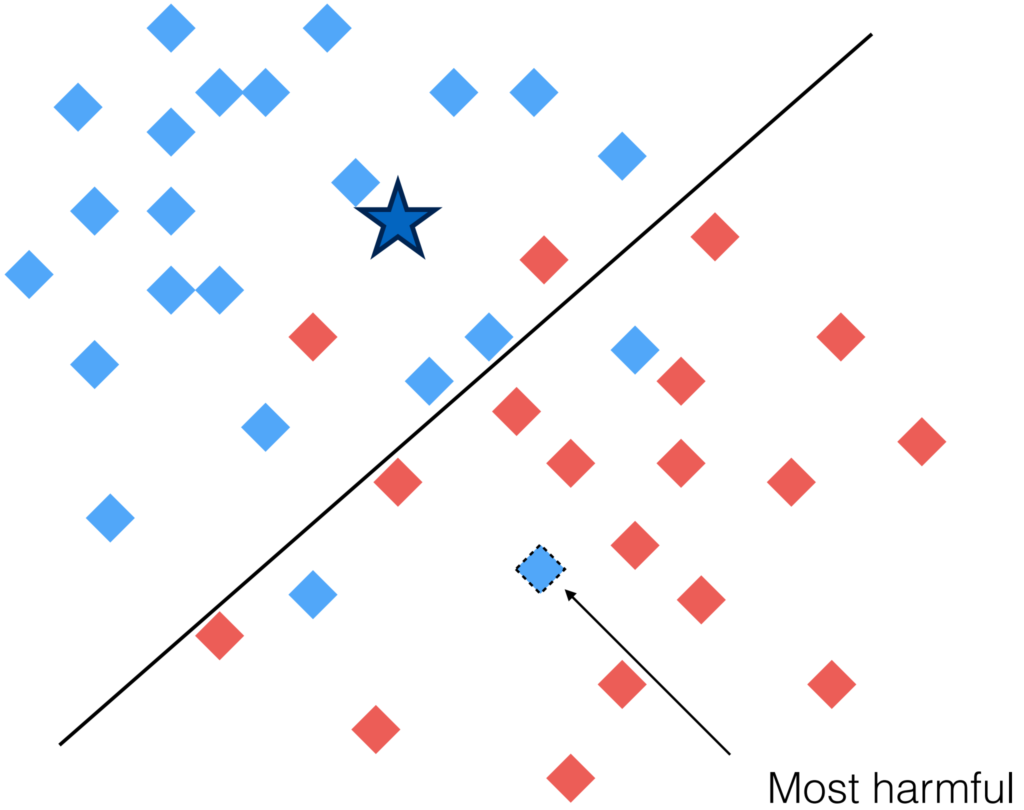
Visualize



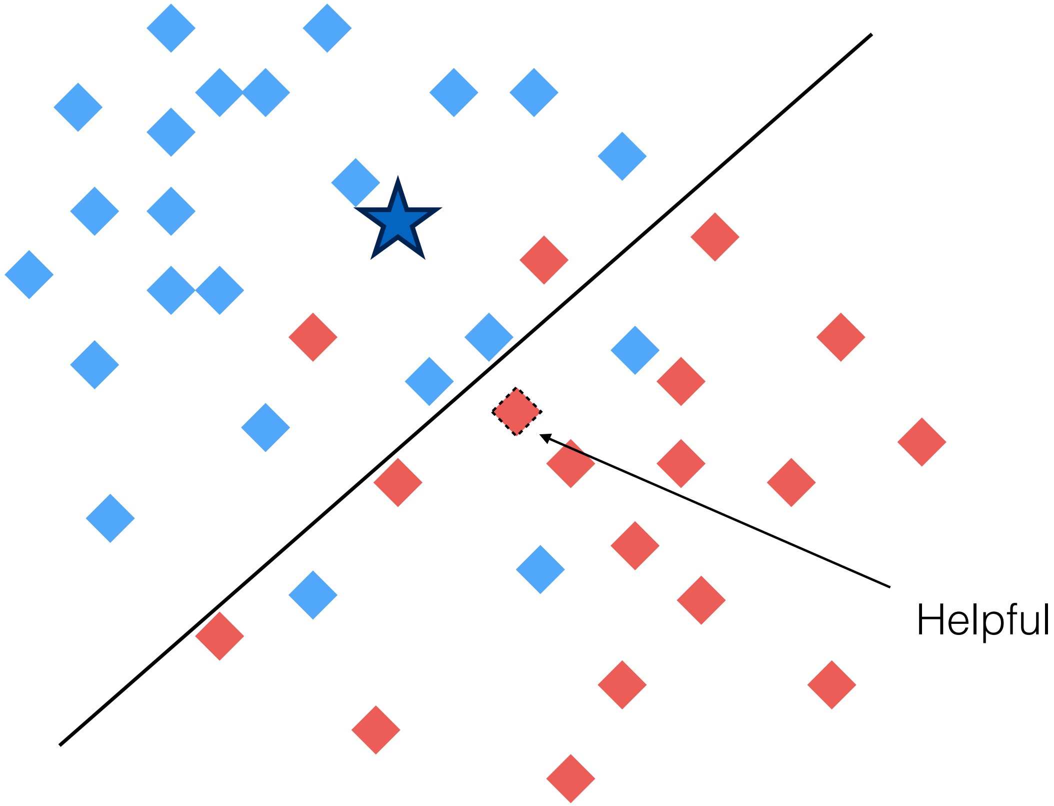
Visualize



Visualize



Visualize



Applied to Deep Models

Training image

Test image

are helpful (of course)

surprisingly helpful

Connect to Input Attribution

- Consider perturbing a training sample z to z_δ
- e.g., $z = (\boldsymbol{x}, y)$, $z_\delta = (\boldsymbol{x}_\delta, y)$
- Training:

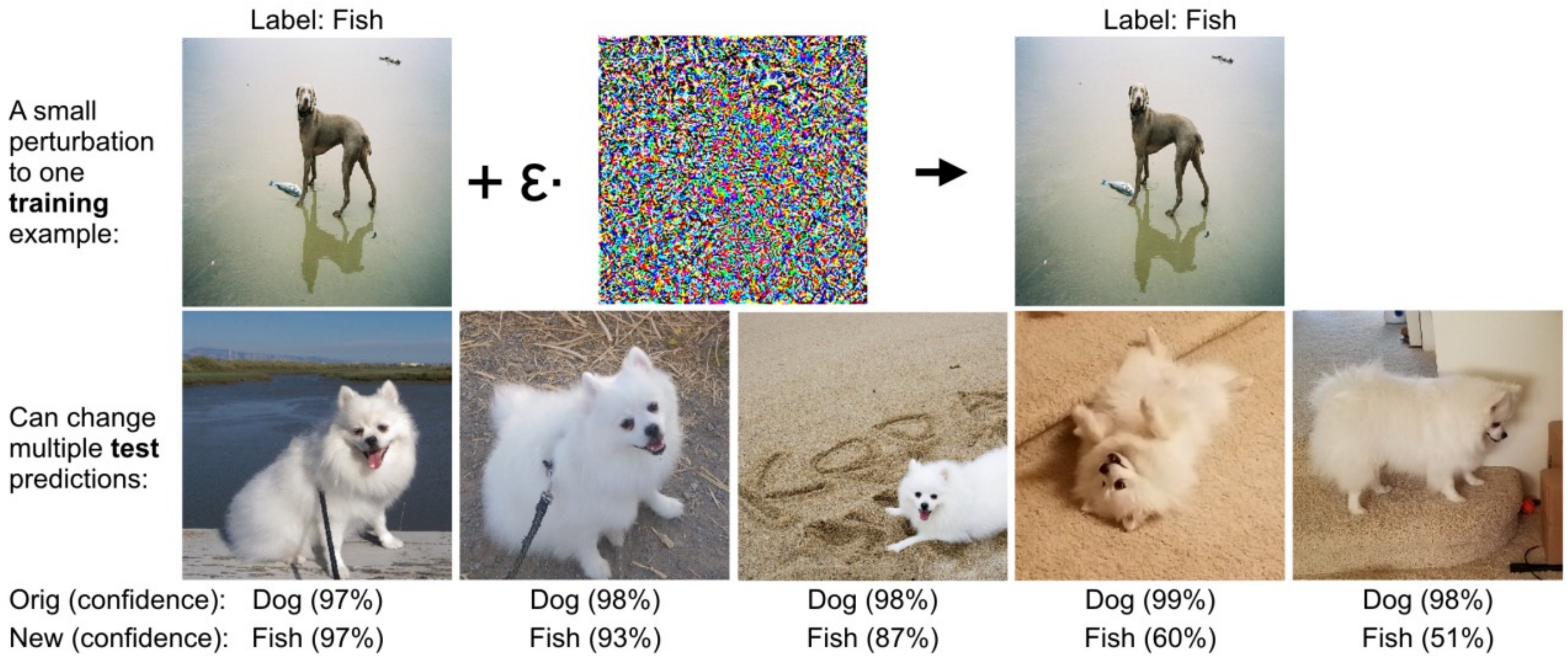
$$\hat{\theta}_\delta = \arg\min_\theta \frac{1}{n}\sum_{i=1}^{n} \ell(\theta; z_i) + \frac{1}{n}[\ell(\theta; z_\delta) - \ell(\theta; z)]$$

- Influence on test sample z, if $\delta \to 0$

$$-\nabla_\theta L(\hat{\theta}_0; z)^T \boldsymbol{H}^{-1} \nabla_{\boldsymbol{x}} \nabla_\theta \ell(\hat{\theta}_0; z) \frac{d\boldsymbol{x}_\delta}{d\delta}$$

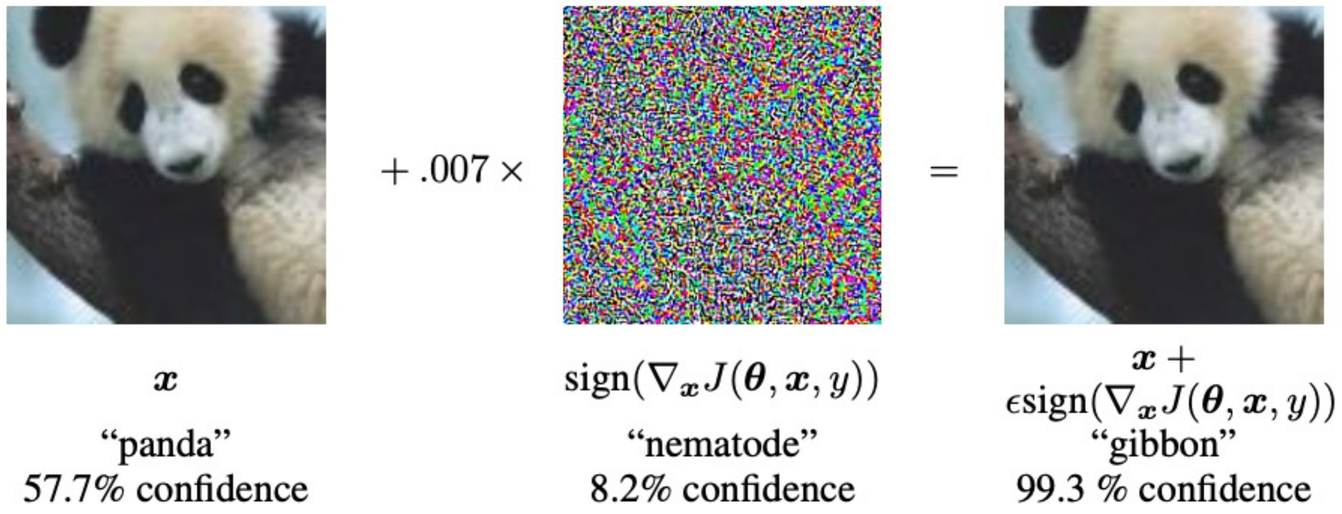
Connect to Input Attribution

$$-\nabla_{\theta} L(\hat{\theta}_0; z)^T \boldsymbol{H}^{-1} \nabla_{\boldsymbol{x}} \nabla_{\theta} \ell(\hat{\theta}_0; z) \cdot \frac{d\boldsymbol{x}_{\delta}}{d\delta}$$



Connect to Adversarial Attack

- Revisit the adversarial example in 1st lecture:



- $\nabla_x J(\theta, \boldsymbol{x}, y)$: the steepest direction that change's network decision
- Influence function traces back to training data!

[Goodfellow et. al, 2015]()