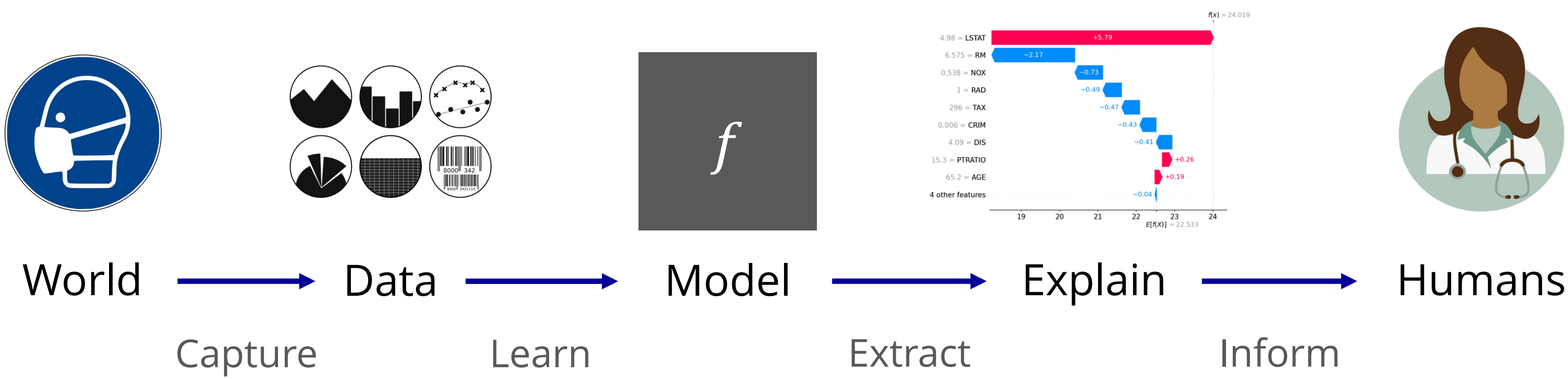
# Inherently interpretable models

CSEP 590B: Explainable AI

Hugh Chen, Ian Covert & Su-In Lee
University of Washington

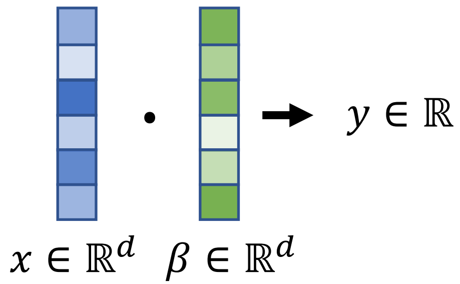
# Post-hoc explanations

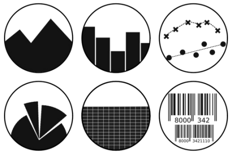



Molnar, "Interpretable machine learning" (2022)

# Inherently interpretable models

Linear model



World → Data → Model → Humans

Capture Learn Extract Inform

# Defining interpretability

- What exactly does “model interpretability” mean?
- Three possible meanings:
    1. Simulatability
    2. Decomposability
    3. Algorithmic transparency

Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery" (2018)

# Simulatability

- Can a human reasonably simulate the model given its parameters and input data?
- No all-purpose definition
  - *Reasonable* is subjective, person-dependent
  - Possibly domain specific

- However, some simple cases:
  - No one can mentally simulate a 50-layer ResNet
  - Most people can simulate a small linear model

# Decomposability

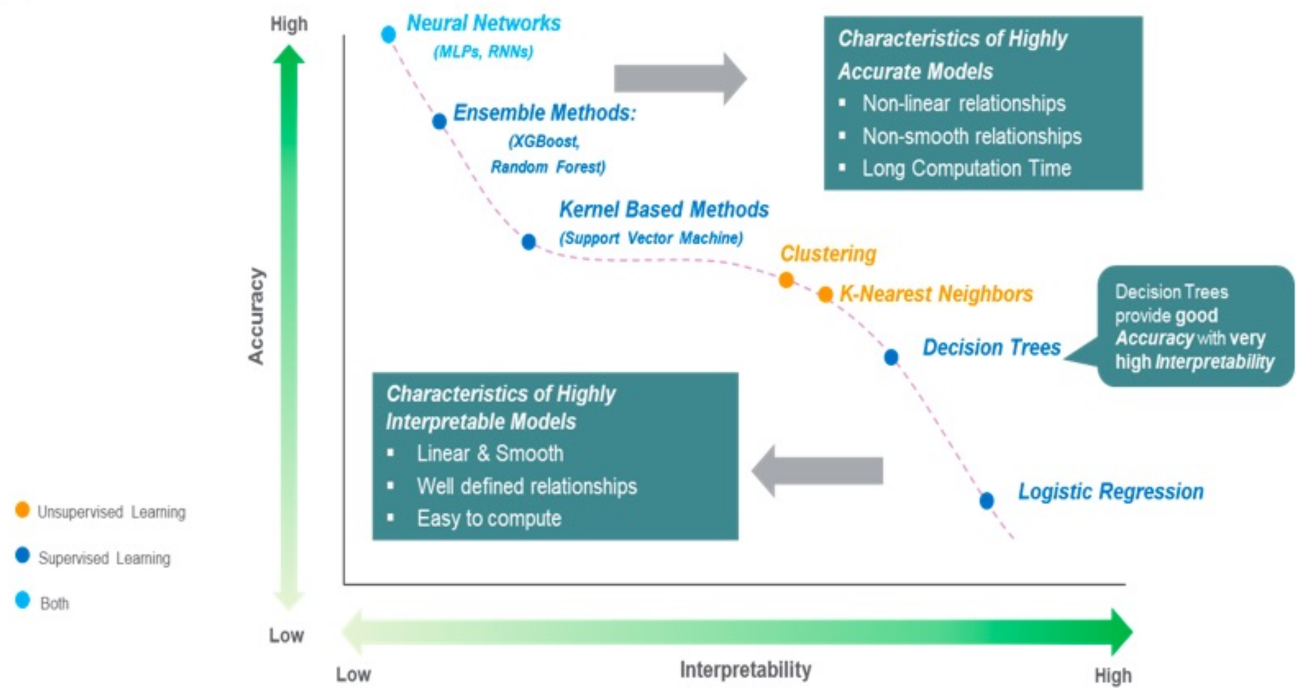
- Does each model component have an intuitive role?
  - Inputs, parameters, operations

- Examples:
  - Each split in a decision tree partitions samples based on a single feature and threshold value
  - Linear model coefficients represent association strength between a feature and the outcome

# Algorithmic transparency

- Can we prove things about the learning algorithm?
  - For example, we've developed a lot of of learning theory for linear models
  - Less for deep models
    - What will the model converge to after training?
    - What types of signals is it likely to use?
    - How does SGD affect under-represented parts of the data distribution, can it affect fairness?

# Why post-hoc explanations?




Rane, "The balance: Accuracy vs. interpretability" (2018)

# Is this tradeoff real?

- For most *structured data* (images, text, audio), neural networks are most accurate
  - After decades of effort with other approaches, many problems now “solved” by DNNs
  - No simple model can match their accuracy
- However, simple models can perform quite well for tabular data
  - E.g., linear/logistic regression, decision lists
  - In this case, less to gain from complex models

# Why this tradeoff?

- Interpretable models tend to be constrained, lack flexibility
- Constrained models can’t represent complex relationships
  - Interpretable models tend to fail in challenging domains, like CV or NLP

# Examples

- Models may be constrained to satisfy linearity, additivity, monotonicity, causality, etc.
- Common examples include:
  - Linear models (linearity)
  - GAMs (limited feature interactions)
  - Decision trees (binary feature splits)

Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead" (2019)

# Caveats

- Even simple models may not be inherently interpretable…
  - If they use engineered features (decomposability)
  - If they use too many features (simulatability)

- Other questions we may ask about the model are not necessarily straightforward
  - What higher-level concepts does the model use?
  - Which training samples influenced the model most?

# Today

- Section 1
  - Introduction
  - Linear regression
  - Generalized additive models (GAMs)
  - Decision trees
- Section 2
  - Class activation maps (CAM)
  - Attention as explanation

# Linear regression

- Linear prediction function:

$$f(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

- Trained by minimizing MSE:

$$\mathcal{L}(\beta) = \sum_{i=1}^{n} \left(y^i - f(x^i)\right)^2$$

# Interpreting a linear model

- Can interpret via learned weights $\hat{\beta}$ and their confidence intervals
  - Quantify feature importance
  - Mentally simulate prediction with new inputs
  - Understand the impact of small changes

# Lasso regression

- Modified approach: find minimal feature set
- Minimize a *regularized* loss function:

$$\mathcal{L}(\beta) = \frac{1}{n}\sum_{i=1}^{n}\left(y^i - f\left(x^i\right)\right)^2 + \lambda\sum_{j=1}^{d}\left|\beta_j\right|$$

- Encourage model to set some weights $\beta_j$ to zero
  - A sparse solution, fewer features are relevant to the prediction

# Ridge regression

- Alternatively, regularize with ridge penalty:

$$\mathcal{L}(\beta) = \frac{1}{n}\sum_{i=1}^{n}\left(y^i - f\left(x^i\right)\right)^2 + \lambda\sum_{j=1}^{d}\beta_j^2$$

- Useful properties, but does not encourage weights to be exactly zero
  - No sparsity, all features remain relevant

# Remarks

- **Pros:**
  - Linear models are easy to interpret, mentally simulate
  - Widely used, fast to train
- **Cons:**
  - Highly constrained, worse predictive performance for some tasks
  - Interpretation is potentially difficult with correlated features

# Today

- Section 1
  - Introduction
  - Linear regression
  - Generalized additive models (GAMs)
  - Decision trees
- Section 2
  - Class activation maps (CAM)
  - Attention as explanation

# GAMs

- Generalized additive models
- Combine non-linear, single-feature models (*shape functions*):

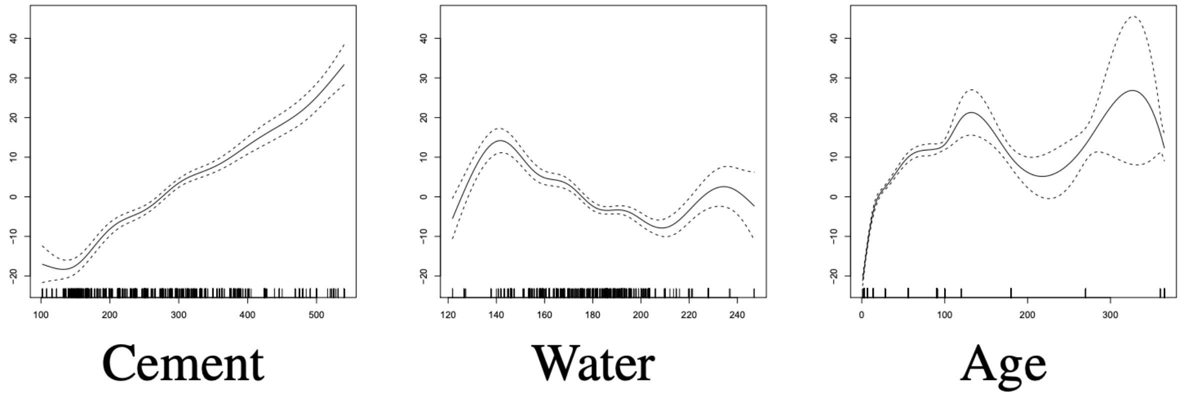
$$f(x) = f_1(x_1) + \cdots + f_d(x_d)$$

- Common options for shape functions:
  - Splines
  - Trees
  - (Linear function = linear regression)

Lou et al., "Intelligible models for classification and regression" (2012)

# Example result

- Relating concrete strength to age and ingredients
  - Splines uncover linear relationship with cement
  - Non-linear relationship with water and age




Lou et al., "Intelligible models for classification and regression" (2012)

# More shape functions

- Piecewise linear curves
  - Ravina et al., "Distilling interpretable models into human-readable code" (2021)
- Deep models
  - Agarwal et al. "Neural additive models: Interpretable machine learning with neural nets" (2020)

# GA²Ms

- GAMs with interaction terms
- Adding pairwise interactions:

$$f(x) = \sum_i f_i(x_i) + \sum_{ij} f_{ij}(x_i, x_j)$$

- Typically, we rank interaction strength for all pairs and decide which to include

Lou et al., "Accurate intelligible models with pairwise interactions" (2013)

# Interactions boost accuracy

| Model | Delta | CompAct | Pole | CalHousing | MSLR10k | Mean |
|---|---|---|---|---|---|---|
| Linear Regression | 0.58±0.01 | 7.92±0.47 | 30.41±0.24 | 7.28±0.80 | 0.76±0.00 | 1.52±0.79 |
| GAM | 0.57±0.02 | 2.74±0.04 | 21.62±0.38 | 5.76±0.55 | 0.75±0.00 | 1.00±0.00 |
| $GA^2M$ Rand | - | - | 11.37±0.38 | - | 0.73±0.00 | - |
| $GA^2M$ Coef | - | - | 11.61±0.43 | - | 0.73±0.00 | - |
| $GA^2M$ Order | - | - | 10.81±0.29 | - | 0.74±0.00 | - |
| $GA^2M$ FAST | **0.55±0.02** | **2.53±0.02** | **10.59±0.35** | **5.00±0.91** | **0.73±0.00** | **0.84±0.20** |
| Random Forests | 0.53±0.19 | 2.45±0.08 | 11.38±1.03 | 4.90±0.81 | 0.71±0.00 | 0.83±0.17 |

**Table 2: RMSE for regression datasets. Each cell contains the mean RMSE ± one standard deviation. Average normalized score is shown in the last column, calculated as relative improvement over GAM.**

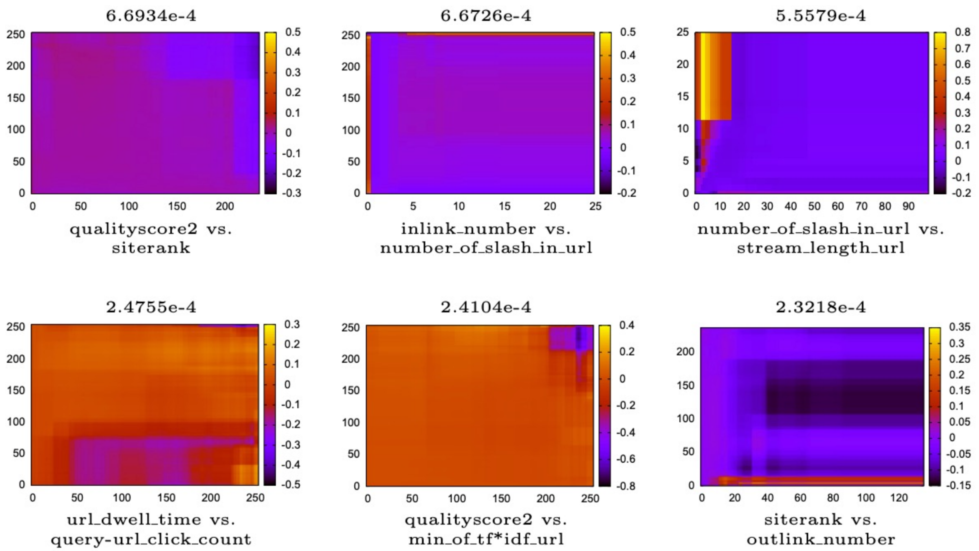
| Model | Spambase | Gisette | Magic | Letter | Physics | Mean |
|---|---|---|---|---|---|---|
| Logistic Regression | 6.22±0.93 | 15.78±3.28 | 17.11±0.08 | 27.54±0.27 | 30.02±0.37 | 1.79±1.25 |
| GAM | 5.09±0.64 | 3.95±0.65 | 14.85±0.28 | 17.84±0.20 | 28.83±0.24 | 1.00±0.00 |
| $GA^2M$ Rand | 5.04±0.52 | 3.53±0.61 | - | - | 28.82±0.25 | - |
| $GA^2M$ Coef | 4.89±0.54 | 3.43±0.55 | - | - | 28.74±0.37 | - |
| $GA^2M$ Order | 4.93±0.65 | 3.08±0.55 | - | - | 28.76±0.34 | - |
| $GA^2M$ FAST | **4.78±0.70** | **2.91±0.38** | **13.88±0.32** | **8.62±0.31** | **28.20±0.18** | **0.81±0.21** |
| Random Forests | 4.76±0.70 | 3.25±0.47 | 12.45±0.64 | 6.16±0.22 | 28.48±0.40 | 0.79±0.26 |

**Table 3: Error rate for classification datasets. Each cell contains the error rate ± one standard deviation. Average normalized score is shown in the last column, calculated as relative improvement over GAM.**

Lou et al., "Accurate intelligible models with pairwise interactions" (2013)

# Example result

- Learning-to-rank dataset, predicting website relevance
  - Interaction effects captured by tree-based GA$^2$M




Lou et al., "Accurate intelligible models with pairwise interactions" (2013)

# Remarks

- GAMs are more flexible than linear models
  - Covers wide class of models with limited feature interactions
  - However, ignores higher-order interactions
- Easier to edit than complex models
  - In some cases, can directly edit model parameters
    - E.g., cap a feature's contribution in recidivism prediction (e.g., number of priors) for policy reasons
    - Manually edit search ranking algorithm

Ravina et al., "Distilling interpretable models into human-readable code" (2021)

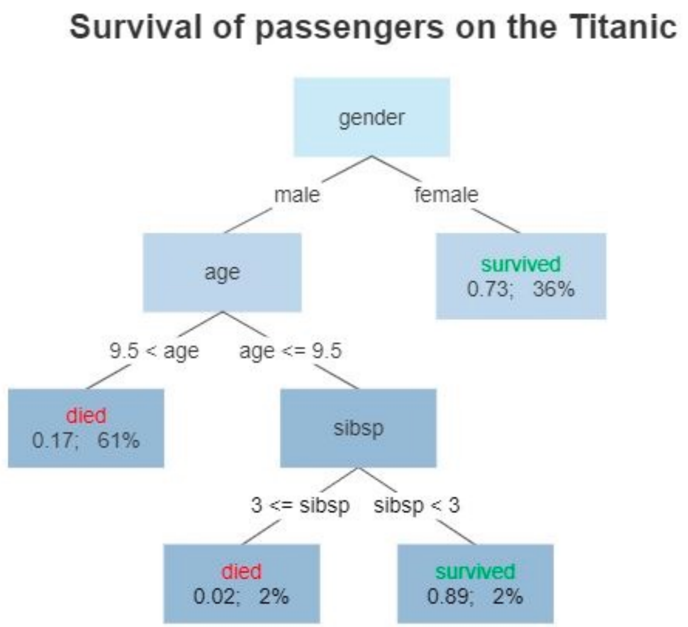
# Today

- Section 1
  - Introduction
  - Linear regression
  - Generalized additive models (GAMs)
  - Decision trees
- Section 2
  - Class activation maps (CAM)
  - Attention as explanation

# Decision trees

- Simple, binary splits
  - Internal node: partition on single feature, threshold
  - Leaf node: predicted outcome for those samples

- Relatively easy to simulate
  - However, can become difficult with more splits



# Decision/rule lists

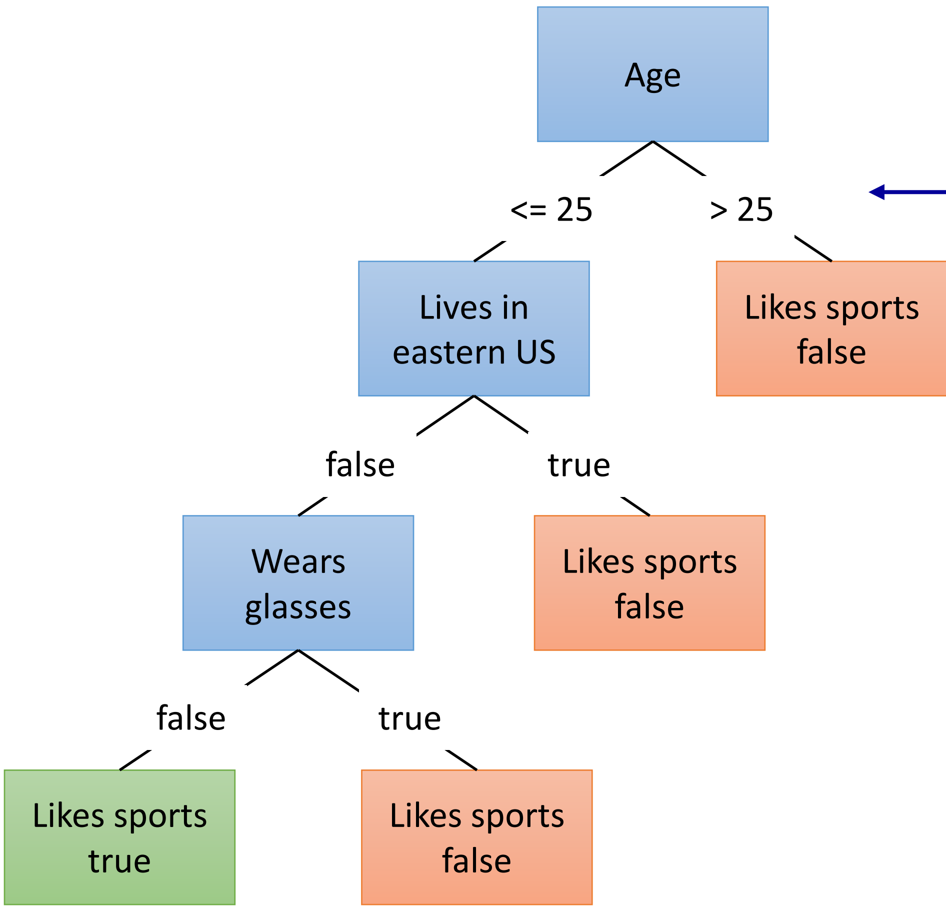
- Decision trees with simplified branching structure
  - For each internal node, one child must be a leaf
  - Like an extended **if – elseif – else** rule

- Decision lists are constrained decision trees
  - Even easier to interpret

Rivest, "Learning decision lists" (1987)

# Example

**Goal:** predict ”likes professional sports”




Internal nodes have at least 1 leaf child

**Subtlety:** by convention, decision lists allow conjunction (AND) of multiple conditions

Simpler structure, but more complex splits than decision trees

# CORELS

- Certifiably optimal rule lists
  - Optimal for a specific class of models on regularized empirical risk
- Branch and bound algorithm to produce an optimal decision list
  - Complex algorithm and data structures to make optimality achievable (vs. standard greedy learning algorithms)
  - Assumptions about the data representation
  - Helps bridge accuracy gap with more flexible models

Angelino et al. "Learning certifiably optimal rule lists for categorical data" (2017)

# Recidivism prediction

Conjunction of conditions

| | | |
|---|---|---|
| IF | age between 18-20 and sex is male | THEN predict arrest (within 2 years) |
| ELSE IF | age between 21-23 and 2-3 prior offenses | THEN predict arrest |
| ELSE IF | more than three priors | THEN predict arrest |
| ELSE | predict no arrest. | |

Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead" (2019)

# Remarks

- CORELS can be slow for very large datasets
  - Greedy learning algorithms are much more efficient
  - Less constrained models (decision trees) offer at least similar performance, possibly better
- Both decision trees and lists are often outperformed by ensemble models
  - Random forests, gradient boosting trees
  - However, these are less interpretable

# Additional desiderata?

- Other potential criteria for inherently interpretable models include:
  - Is it easy to identify **feature adjustments** to achieve a different outcome?
  - Is it easy to determine impact of withholding a **feature's information**?
  - Is it easy to change your **model** to fix undesirable behavior?
  - Can you determine which **data points** influenced the model's prediction?

- Some of these are possible with previously discussed models, but others are not

# Today

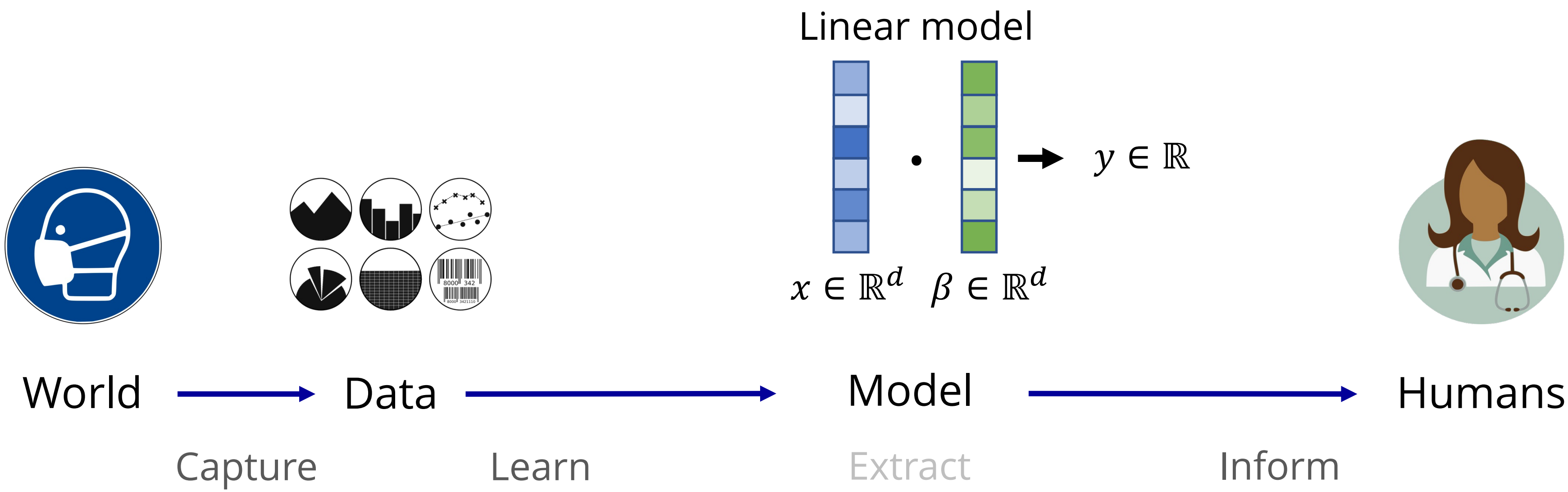
- Section 1
  - Introduction
  - Linear regression
  - Generalized additive models (GAMs)
  - Decision trees
  - **10 min break**
- Section 2
  - Class activation maps (CAM)
  - Attention as explanation

# Interpretable complex models

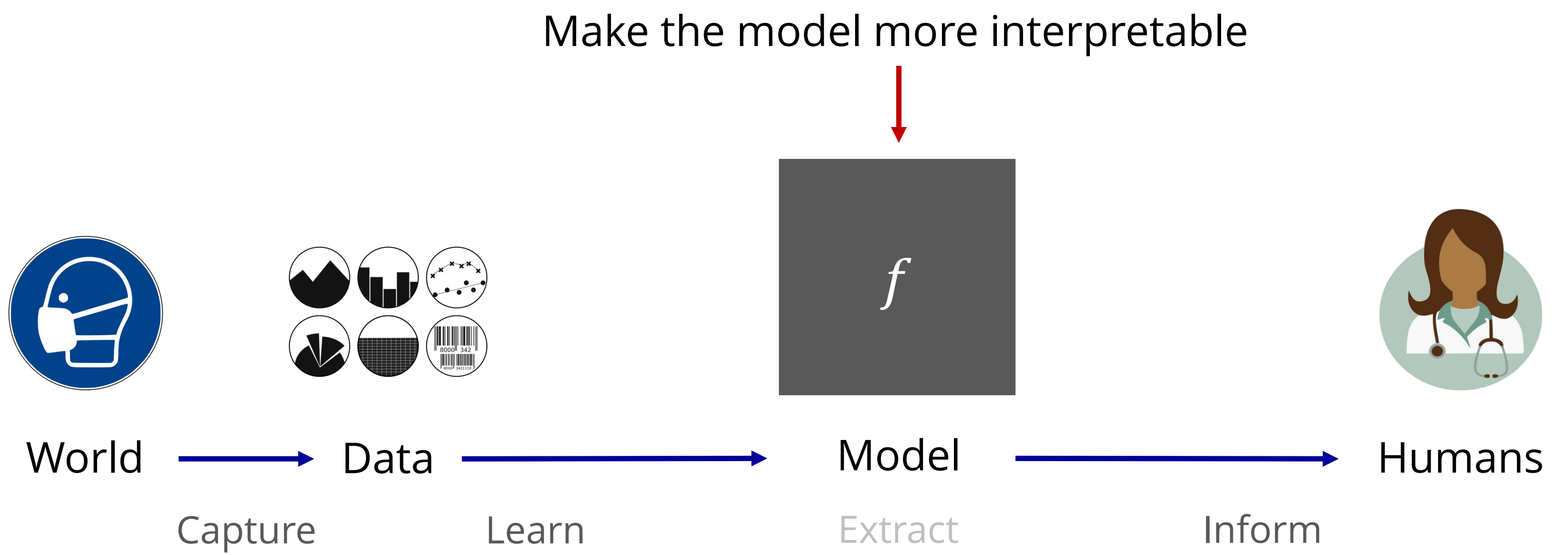
CSEP 590B: Explainable AI

Hugh Chen, Ian Covert & Su-In Lee
University of Washington

# Inherently interpretable models (previously)



# Interpretable complex models



# Interpretable complex models (cont.)

- For inherently complex models (e.g., DNNs), we can make them *more* interpretable

- Today: two deep learning architectures that enable interpretability
  1. CNNs with global average pooling
     - Class activation maps (CAM)
  2. Transformers based on self-attention
     - Attention-based explanations

# Today

- Section 1
  - Introduction
  - Linear regression
  - Generalized additive models (GAMs)
  - Decision trees
- Section 2
  - Class activation maps (CAM)
  - Attention as explanation

# Class activation maps

- Built-in feature attribution for CNNs with specific output layers
  - Global average pooling followed by linear layer


Zhou et al., "Learning deep features for discriminative localization" (2016)

# CNN architecture refresher

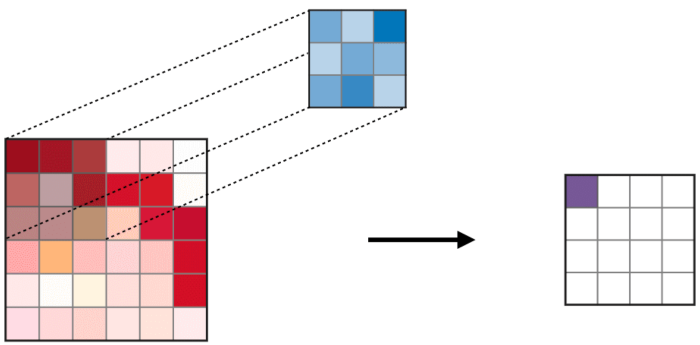
- Some common CNN architectures include AlexNet, VGG, ResNet, DenseNet

- Networks typically consist of:
  - Convolutional layers
  - Max pooling layers
  - Fully-connected layers

# Layer types

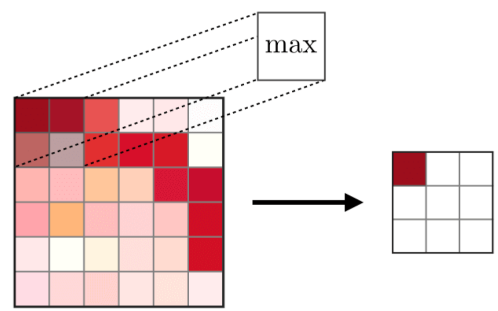
**Convolutional layers**

- Apply learned kernel to each position
- Shared + localized feature extraction



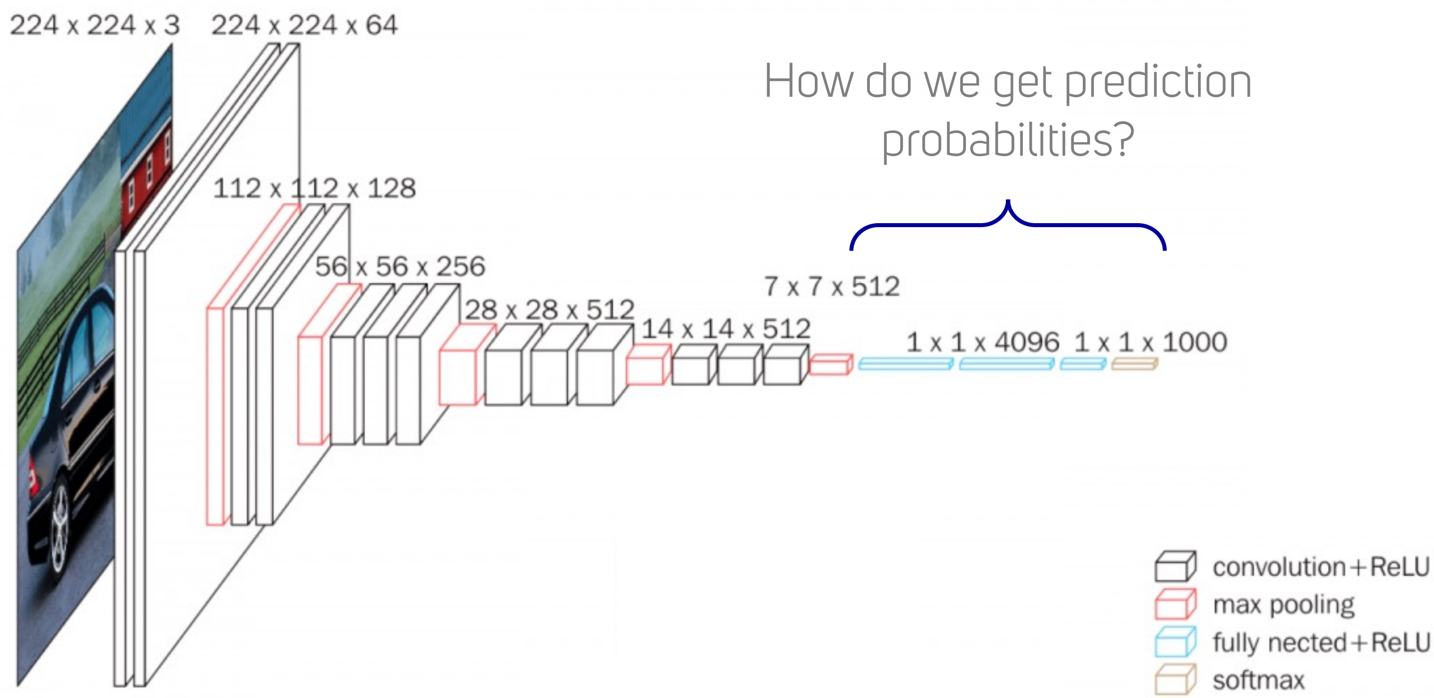
**Max-pooling layers**

- Calculate max value within sliding window
- Downsample to lower resolution




Amidi & Amidi, "Convolutional neural networks cheat sheet"

# VGG architecture




How do we get prediction probabilities?

Convolutions and max pooling

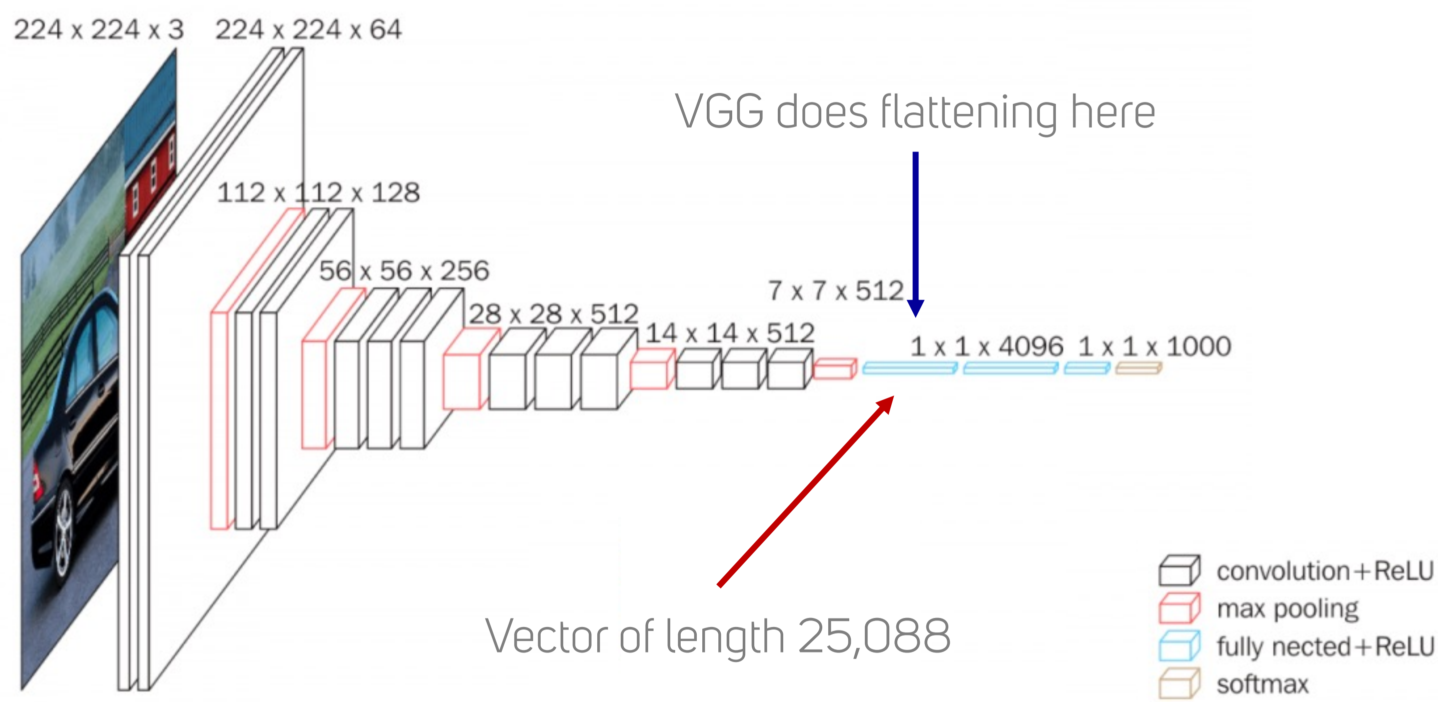
# CNN output layers

- Conv + max pool output has extra dimensions
  - Tensor with shape $h \times w \times c$ (height, width, channels)
  - We need probability vector of length $M$ (# classes)

- Options:
  1. Flatten into vector of length $hwc$
  2. Pool along spatial dimensions: vector of length $c$

- Then, apply fully-connected and softmax layer(s)

# VGG architecture




Next FC layer has *100M* parameters!

# Global average pooling

- Calculate spatial average of last layer features
  - Let $A \in \mathbb{R}^{h \times w \times c}$ be last tensor
  - $A_{ij}^k$ is value at position $(i, j)$, channel $k$
  - Calculate $\bar{A}_k = \frac{1}{hw} \sum_{ij} A_{ij}^k$
- Fewer learnable parameters, less overfitting
- GAP used in many popular architectures
  - E.g., ResNet, DenseNet

Lin et al., "Network in network" (2013)

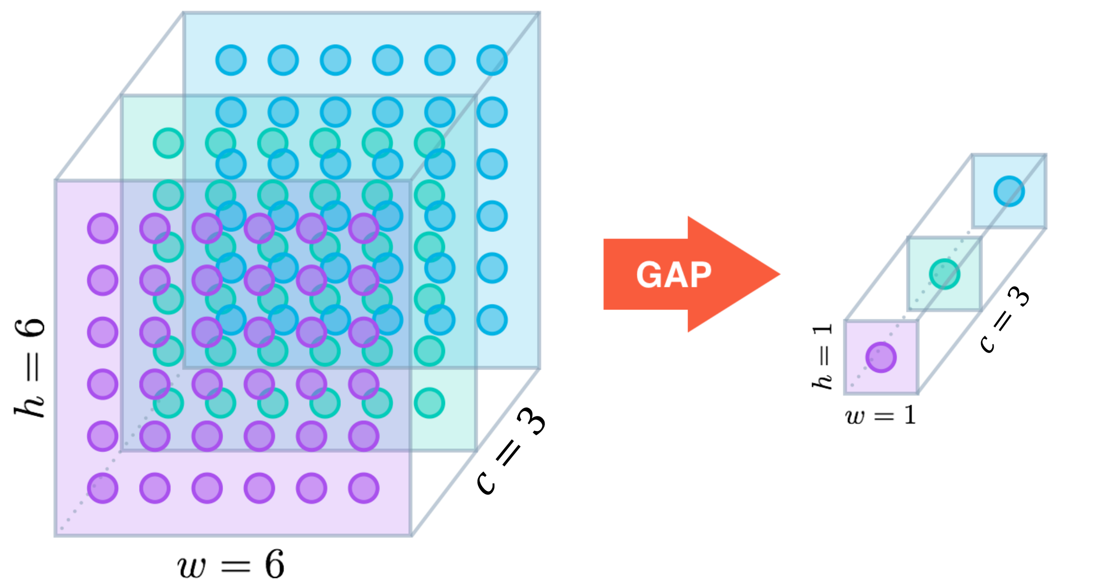
# Putting it together

- Conv + max pool to get $A \in \mathbb{R}^{h \times w \times c}$
- GAP to get $\bar{A} \in \mathbb{R}^{c}$
- Fully-connected layer to get logits $z \in \mathbb{R}^{M}$:

$$z_y = \sum_{k=1}^{c} w_k^y \cdot \bar{A}_k$$

- Finally, softmax turns each $z_y$ into a probability

# Putting it together (cont.)

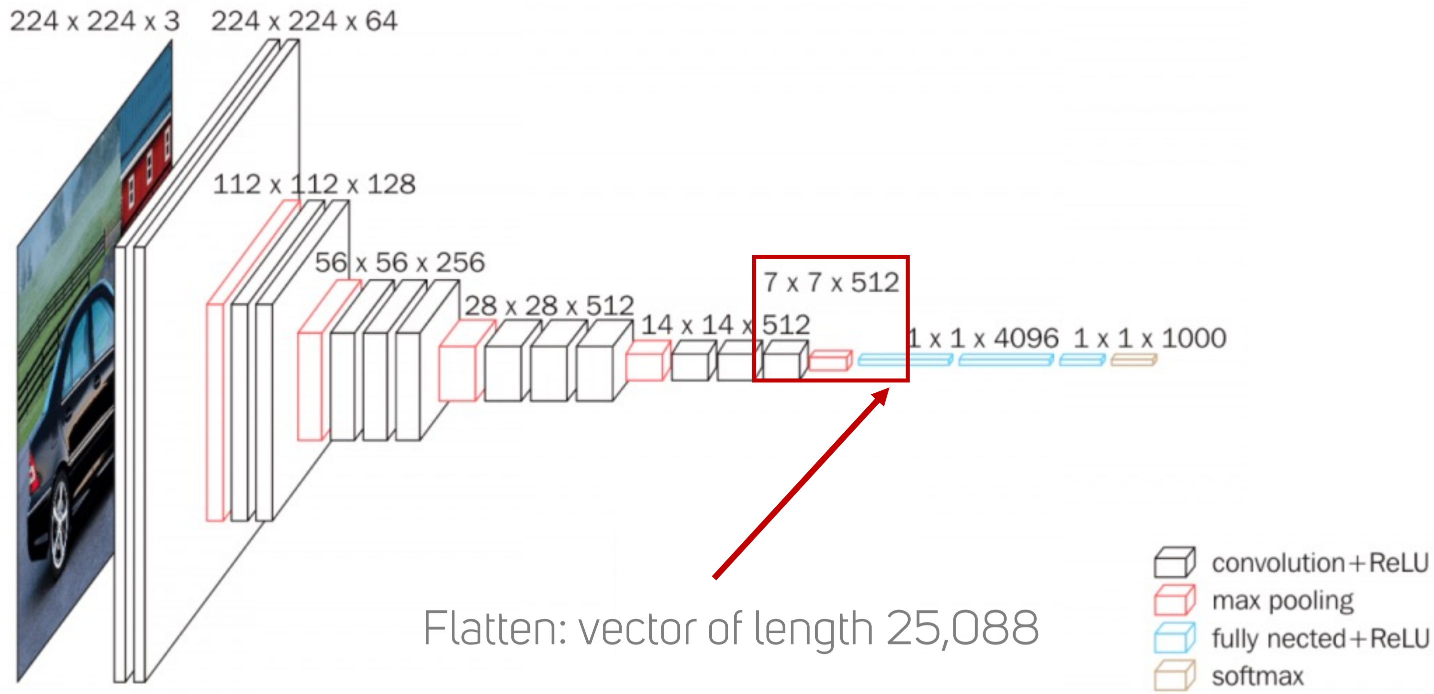



$A \in \mathbb{R}^{h \times w \times c}$

$\bar{A} \in \mathbb{R}^{c}$

Cook, "Global average pooling layers for object localization" (2017)

# Applied to final tensor $A$




Flatten: vector of length 25,088

GAP: vector of length 512

# Class activation maps (CAM)

- **Idea:** view GAP + FC layer as averaging separate predictions from each spatial position

$$z_y = \sum_{k=1}^{c} w_k^y \cdot \frac{1}{hw} \sum_{ij} A_{ij}^k \quad \longleftarrow \bar{A}_k$$

$$= \frac{1}{hw} \sum_{ij} \boxed{\sum_{k=1}^{c} w_k^y A_{ij}^k} \quad \longleftarrow$$

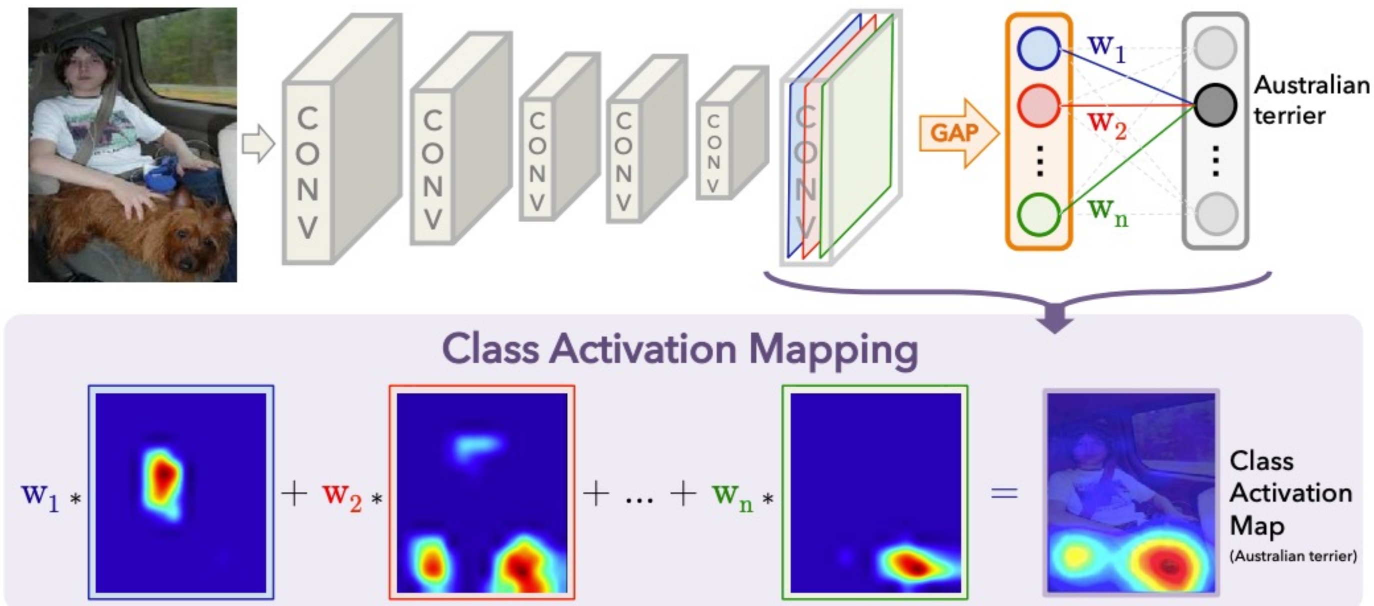
Swap order of summation

- Define importance for class $y$ as:

$$a_{ij} = \sum_{k=1}^{c} w_k^y A_{ij}^k \quad \longleftarrow$$

Like applying FC layer separately at each position

# Alternative view

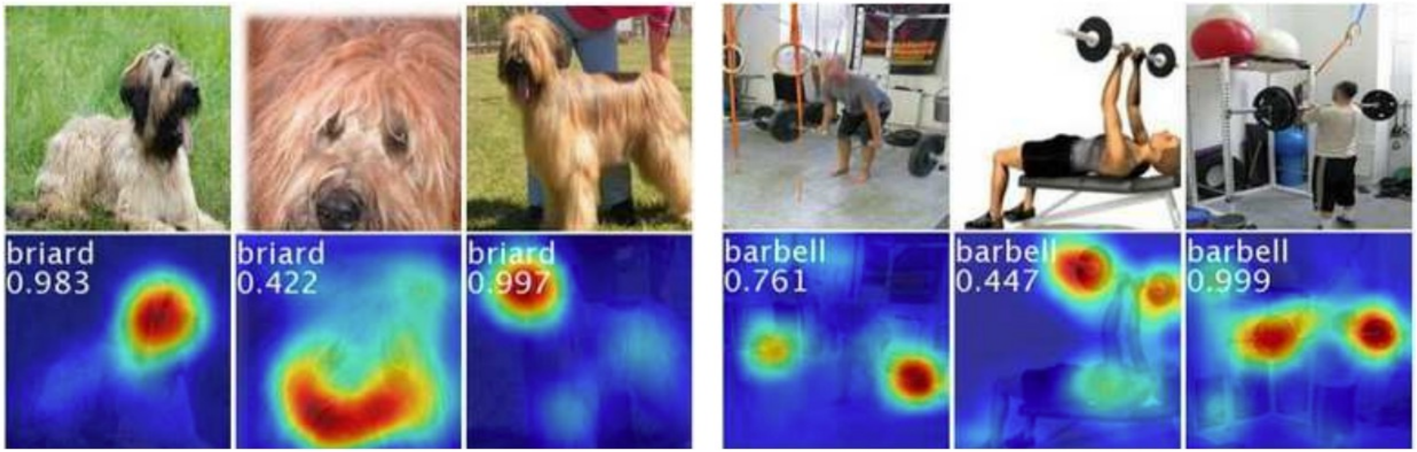



Use final layer weights to combine convolutional features

CAM results

Zhou et al., "Learning deep features for discriminative localization” (2016)

# Qualitative evaluation




Zhou et al., "Learning deep features for discriminative localization" (2016)

# Localization with CAM

Table 3. Localization error on the ILSVRC test set for various weakly- and fully- supervised methods.

| Method | supervision | top-5 test error |
|---|---|---|
| GoogLeNet-GAP (heuristics) | weakly | **37.1** |
| GoogLeNet-GAP | weakly | 42.9 |
| Backprop [22] | weakly | 46.4 |
| GoogLeNet [24] | full | 26.7 |
| OverFeat [21] | full | 29.9 |
| AlexNet [24] | full | 34.2 |

Simonyan et al., 2013
(last time)

Zhou et al., "Learning deep features for discriminative localization" (2016)

# Relationship with GradCAM

- Recall that GradCAM defines feature importance as

$$a_{ij} = \sum_k \alpha_k^y A_{ij}^k$$

where we have:

$w_k$ in CAM

$$\alpha_k^y = \frac{1}{wh} \sum_{ij} \frac{\partial f_y}{A_{ij}^k}$$

- **Result:** GradCAM = CAM when we use GAP + FC
- GradCAM allows nonlinearities after GAP, or no GAP

# Spatial locality assumption

- CAM/GradCAM assume internal feature maps correspond to original input space
  - Roughly true due to convolutional structure
  - However, may not hold for later layers in very deep networks
  - GradCAM can operate in intermediate layers, where spatial locality is better preserved

# CAM remarks

- Strong results, particularly in object localization
- Can only be computed for specific architectures (when using GAP + FC)
- Assumes spatial locality in final layer, which may not hold for very deep models

# Today

- Section 1
  - Introduction
  - Linear regression
  - Generalized additive models (GAMs)
  - Decision trees
- Section 2
  - Class activation maps (CAM)
  - Attention as explanation

# Attention

- A key component in some recent deep learning architectures

- **Human attention:** focusing on certain stimuli around us (visual, auditory, etc.)

- **Attention in DL:** using small portion of features to generate a prediction
  - Typically used at hidden layers with internal features
  - Features that get no attention are set to zero

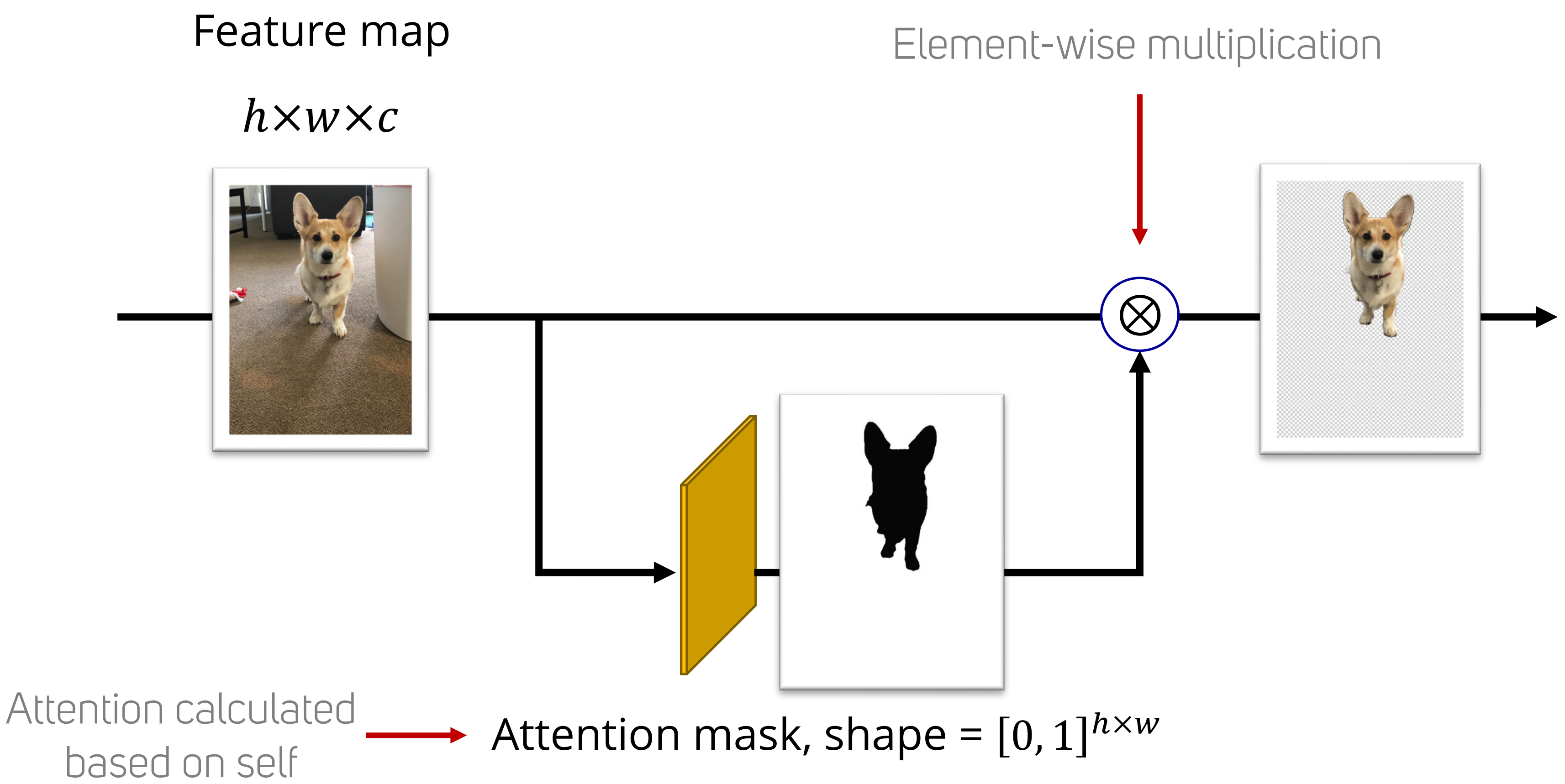
# Attention in DL

- A core component of modern NLP models
  - Increasingly popular for vision as well

- Hard vs. soft attention
  - Multiply by exactly zero, or approximately zero?
  - The latter is easier to learn via gradient descent

- How does it work?
  - How are attention values computed?
  - How are they used?

# Self-attention example



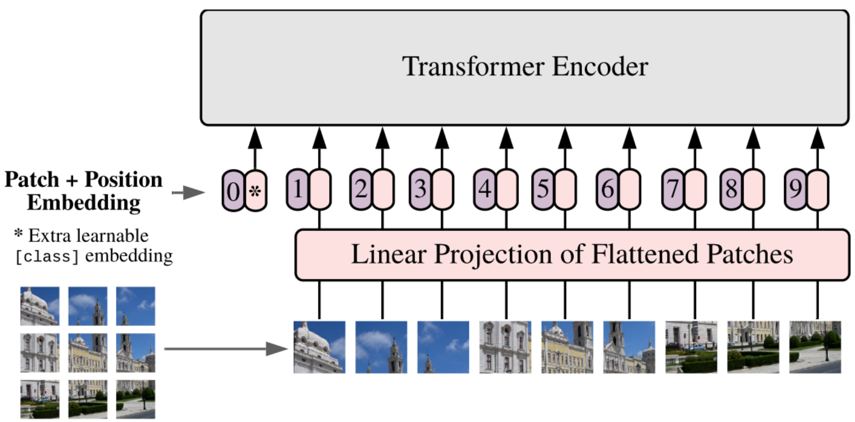
# Self-attention

- Some usage in CNNs
  - Jetley et al., "Learn to pay attention" (2018)
- Mostly used in **transformers**
  - Popularized in machine translation, now SOTA in basically all NLP tasks
    - Language modeling (GPT-3), masked language modeling (BERT)
  - Protein modeling (e.g., AlphaFold)
  - Vision transformers (ViTs)

Vaswani et al., "Attention is all you need" (2017)

# Case study: ViTs

- An alternative to CNNs, and currently a hot research area
- Built on self-attention operation




Passed to transformer encoder

Image patches become "tokens"

(in NLP, tokens are words or words parts)

Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale" (2020)

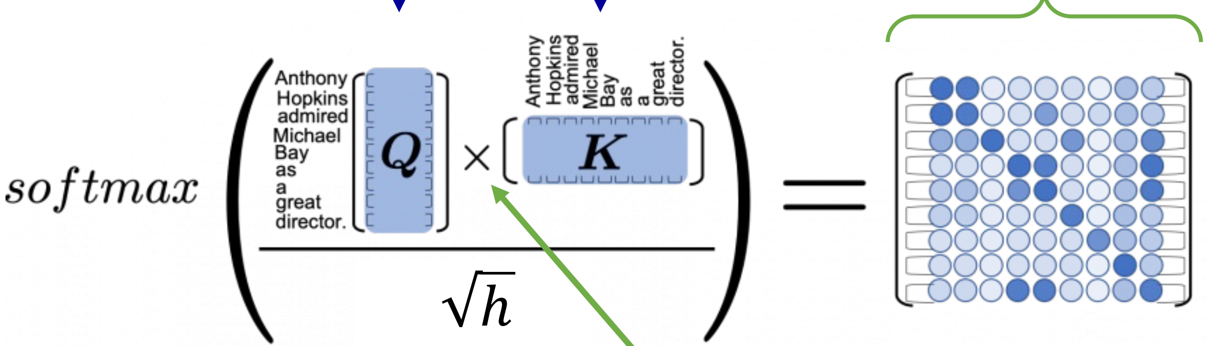
# Self-attention operations

- Sequence of operations at each layer
  - Every token gets a **query**, **key**, and **value** vector
  - Use **query** and **key** to determine relevance for each token pair $(i, j)$
  - Normalize relevance to get **attention** values
  - Use **attention** to average **value** vectors for each token
    - Each token in the next layer becomes weighted sum of all previous token values
    - Attention controls weight for each token

# Attention matrix

One query, key vector per token

Attention matrix, rows sum to 1




Dot product for all pairs

Normalization constant (not that important)

Tamura, "Multi-head attention mechanism: queries keys and values, over and over again" (2021)

# Mathematical notation

- Let embeddings for $d$ tokens be $z \in \mathbb{R}^{d \times h_e}$
- Let parameters be $W_q, W_k, W_v \in \mathbb{R}^{h_e \times h_a}$
- Calculate **queries**, **keys**, **values** $\in \mathbb{R}^{d \times h_a}$ as:

$$[Q, K, V] = [zW_q, zW_k, zW_v]$$

Per-token query, key, value vectors, size $h_a$

- Calculate **attention** values $A \in \mathbb{R}^{d \times d}$ as:

Dot product between query, key for all pairs

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{h}}\right)$$

Softmax applied along second dimension

- Calculate self-attention output $\mathrm{SA}(z) \in \mathbb{R}^{d \times h_a}$ as:

$$\mathrm{SA}(z) = AV$$

Each token becomes weighted sum of value vectors

# Complete architecture

- ViTs are composed of many self-attention layers
  - In reality, they use multi-head self-attention
  - The same operations, but performed in parallel
- In addition…
  - Layer norm
  - Fully-connected layers in between
  - Possible residual connections between layers
  - Output calculated using class token

Okay to ignore for now

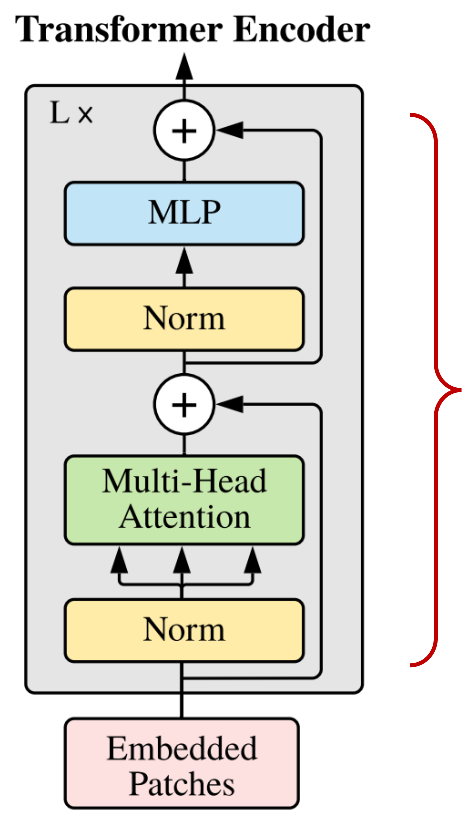
- We'll just focus on self-attention

# Complete architecture (cont.)

Several of these layers ($\mathbf{L}$)

Self-attention operation

Denote attention at layer $b$ as $A^{(b)}$



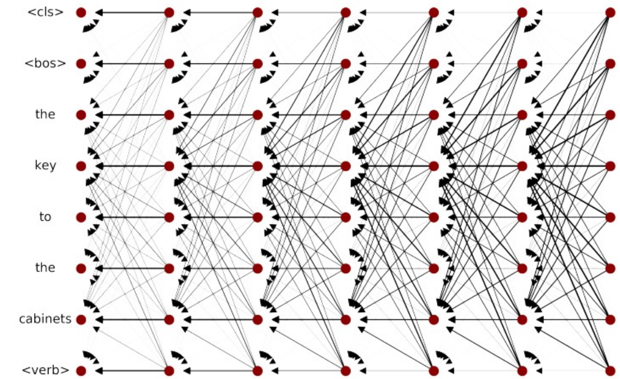

Other operations

# Raw attention

- **Idea:** define important features as those that receive most attention

- Sounds reasonable, but attention is calculated at every layer and for every pair of tokens
- Simple approach:
  - Examine a single layer (e.g., last layer)
  - Examine attention directed to the *class token*
    - Special token that's ultimately used to make predictions
    - Extract a single row of $A^{(L)}$

# Attention rollout




- **Problem:** information mixes between tokens at each layer

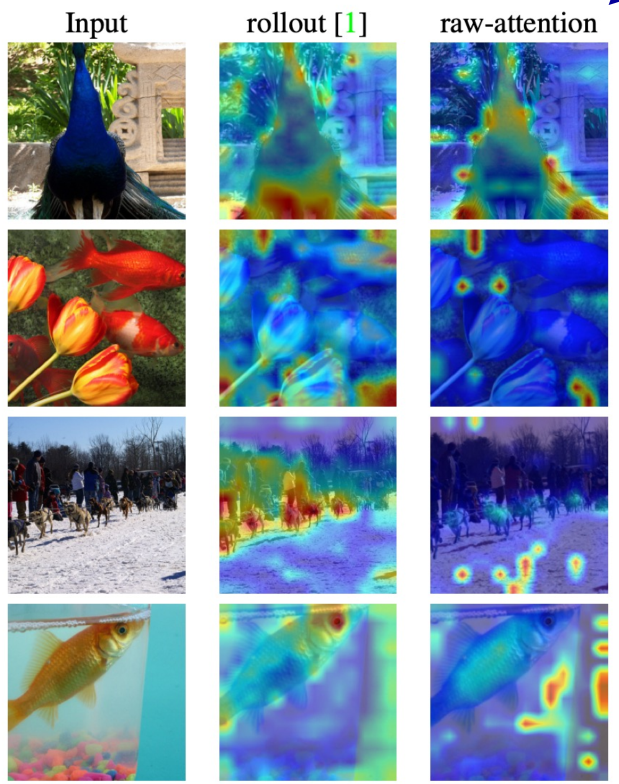
- **Idea:** treat attention as a graph, examine flow
  - Add identity to each attention matrix, $\hat{A}^{(b)} = A^{(b)} + I$
  - Calculate the product, $\text{rollout} = \hat{A}^{(1)} \cdot \hat{A}^{(2)} \ldots \cdot \hat{A}^{(L)}$
  - Extract a single row of the rollout matrix, again for class token

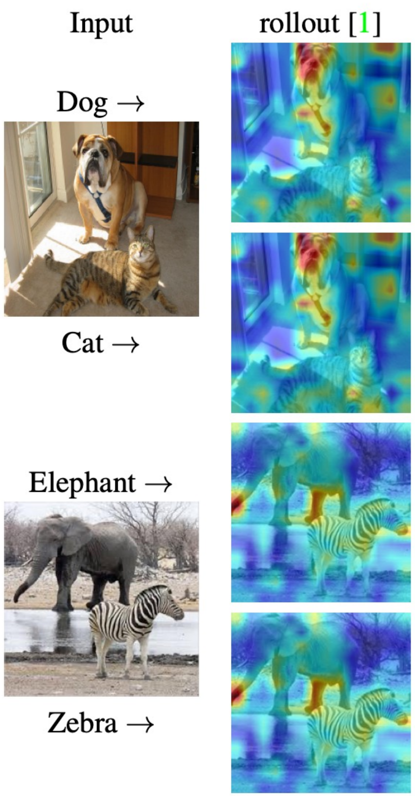
Abnar & Zuidema, "Quantifying attention flow in transformers" (2020)

# Examples

Unconvincing results (qualitatively)

Not class-specific




Chefer et al., "Transformer interpretability beyond attention visualization" (2021)

# Other examples

- More papers interpreting transformers via attention
  - Clark et al., "What does BERT look at? An analysis of BERT's attention" (2019)
  - Rogers et al., "A primer in BERTology: What we know about how BERT works" (2020)
  - Vig et al., "BERTology meets biology: interpreting attention in protein language models" (2020)

# Attention skepticism

- Is attention a valid approach to understand feature importance?
  - No guarantee that attention functions how we envision (like human attention)
  - Overlooks other operations in transformers

- Several papers on this topic
  - Serrano & Smith, "Is attention interpretable?" (2019)
  - Jain & Wallace, "Attention is not explanation" (2019)
  - Wiegreffe & Pinter, "Attention is not not explanation" (2019)

# Remarks

- **Pros:**
  - Attention is calculated automatically for the prediction, minimal overhead
  - Clear meaning: weight for each token in self-attention operation

- **Cons:**
  - Not obvious how to aggregate across attention heads, layers, and pair-wise interactions
  - Reductive, ignores other important operations
  - Weak results in XAI metrics (see Chefer et al., 2021)

# Summary

- Global average pooling and self-attention were both introduced to improve predictive performance
- Later used to make models more interpretable
- Other approaches explicitly aim to make deep learning models more interpretable
  - Chen et al., "This looks like that: deep learning for interpretable image recognition" (2019)
  - Wang et al., "Shapley explanation networks" (2021)