Biological Applications of Deep Learning
Lecture 11

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
January 11, 2023

Contents Today

- DiseaseCapsule
  - Predicting ALS disease status from genotype profiles
  - Using capsule networks for
    - enhanced prediction
    - enhanced interpretation and
    - economic training data usage
- Attention Mechanisms I
  - Basic Idea: Queries, Keys and Values
  - Nadaraya-Watson Regression
  - Attention Scoring Functions

Disease Capsule

Reference

- X. Luo, X. Kang, A. Schönhuth
  Predicting the prevalence of complex genetic diseases from individual genotype profiles using capsule networks
  Nature Machine Intelligence, to appear

Reminder: Learning the Genetic Architecture

The Genetic Architecture of ALS

Definition

Let X be all people, represented by their genotype profiles.

The *genetic architecture* f_{ALS} of ALS is a function

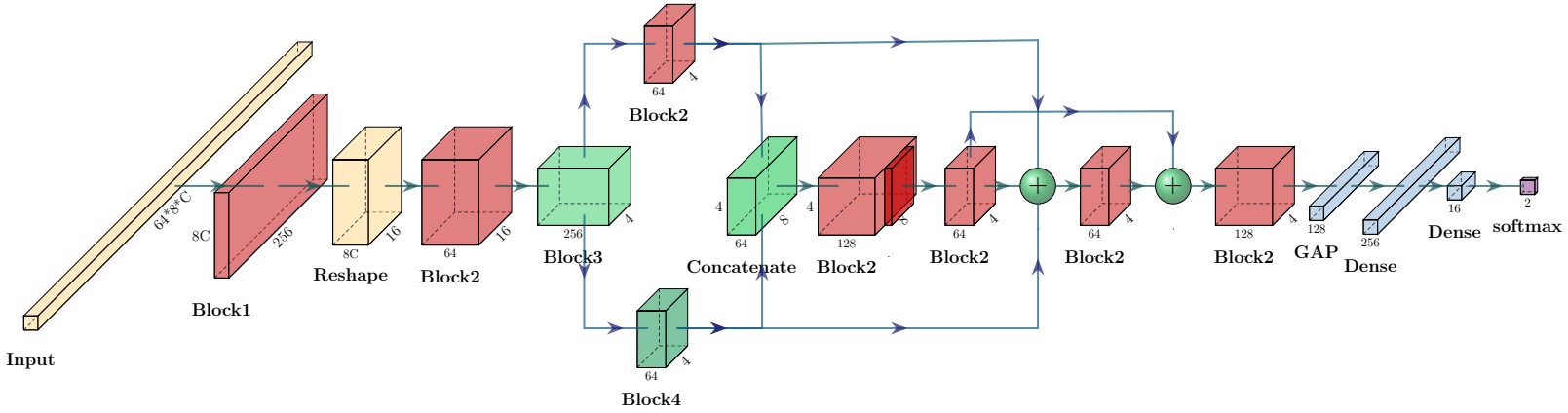
$$f_{\text{ALS}} : X \longrightarrow \{0, 1\}$$

where

$$f(x) = \begin{cases} 1 & x \text{ affected by ALS} \\ 0 & \text{otherwise} \end{cases}$$

Prior Work: Convolutional Neural Network

[Yin et al, 2019]

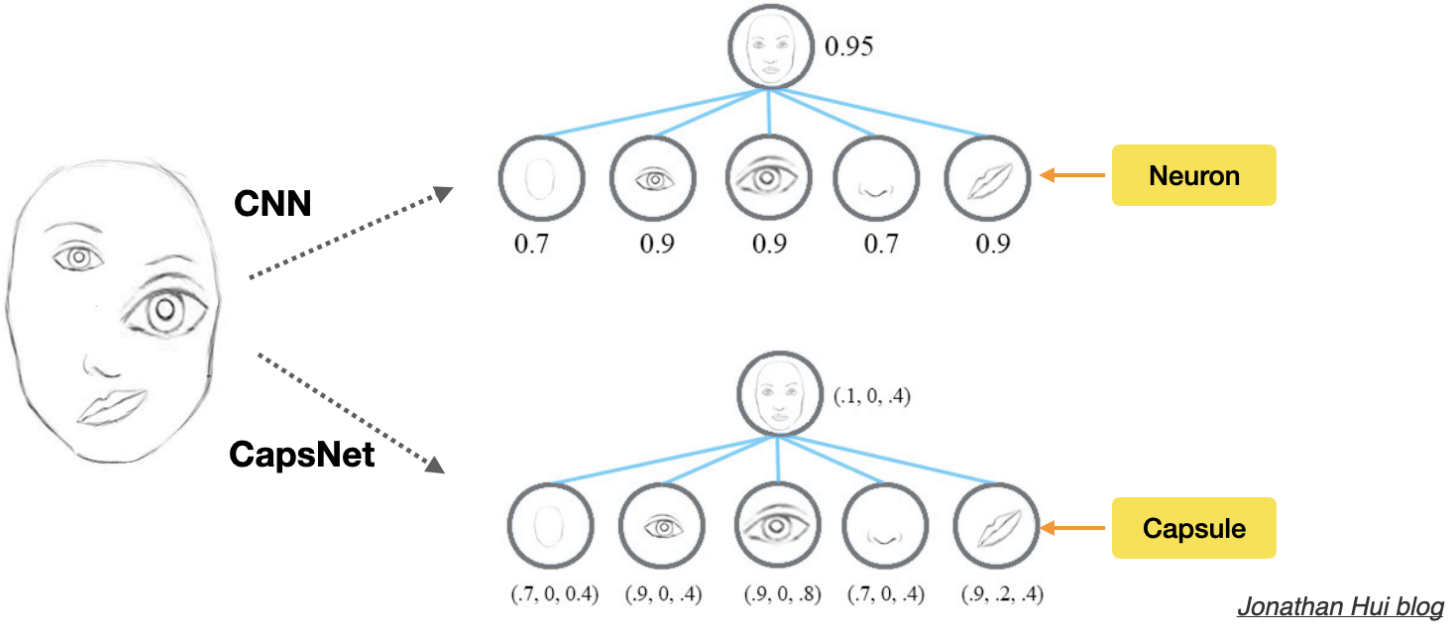


ALS-Net

- *Advantage:* Good accuracy (76.9%) in prediction using 4 chromosomes
- *Disadvantages:*
  - Black box character ☞ interpretation of results impossible
  - Requires large amounts of training data
  - Whole genome input leads to overfitting
  - May get confused when combining effects

Capsule Networks – Motivation

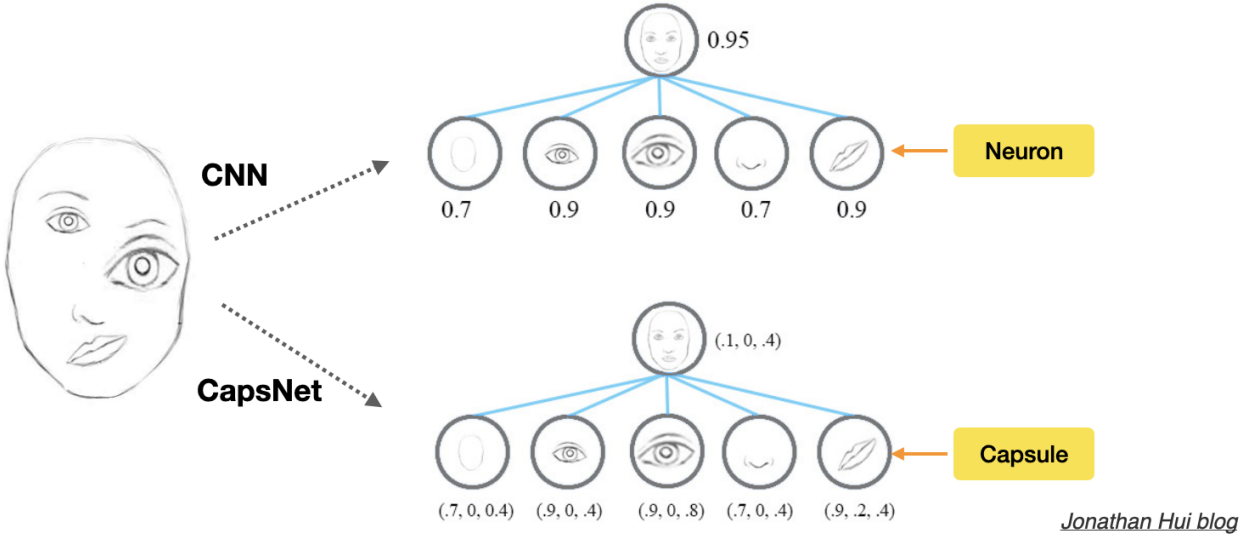
Capsule Networks: Theoretical Advantages



Jonathan Hui blog

- Capsules = vector style neurons; can handle distortions and overlaps
- Point out natural ways to interpret results ☞ *break open black box*
- Require sufficiently less training data and are more accurate

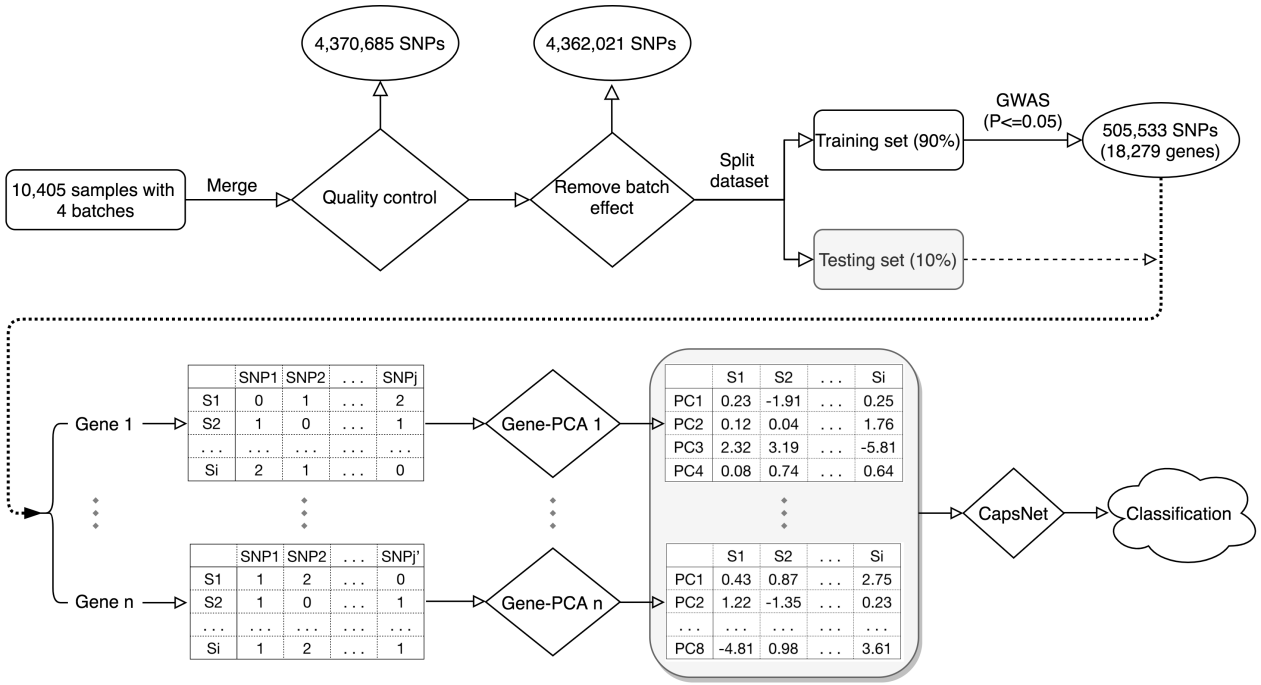
Capsule Networks: Motivation



- *Goal:* enable whole genome input processing *without confusion*
  - Map spatially local structures intrinsically present across genomes
  - Map hierarchical structures corresponding to protein complexes, pathways, processes, compartments
- Reasonable, meaningful interpretation of results
- Less training data ☞ prevents generation of massive cohorts

Methods

Workflow



Feature selection:

- Standard routines for discarding irrelevant SNP's
- *Novel: Gene-PCA reduces dimensionality* while preserving non-linearity

Gene-PCA: Protocol I

- Let $M \approx 20\,000$ be the number of genes of the human genome
- One can assign a region C_i of the genome to each gene $g_i, i = 1, ..., M$
  - The C_i are supposed to be non-overlapping
  - Together, the C_i are supposed to span the entire genome
- Let N_i be the number of polymorphic sites contained in C_i
  - If N is the number of polymorphic sites overall, then

$$N = \sum_{i=1}^{M} N_i$$

Gene-PCA: Protocol II

- Let N_i be the number of polymorphic sites contained in gene region C_i
  - If N is the number of polymorphic sites overall, then $N = \sum_{i=1}^{M} N_i$
- Let $X \in \{0, 1, 2\}^N$ be an individual genotype profile
- Let

$$X(i) \in \{0, 1, 2\}^{N_i} \quad \text{where} \quad X(i)[j] = X[\sum_{k<i} N_k + j], j = 1, ..., N_i \tag{1}$$

be the vector whose entries correspond to the genotypes referring to the polymorphic sites in the contiguous region C_i

Gene-PCA: Protocol III

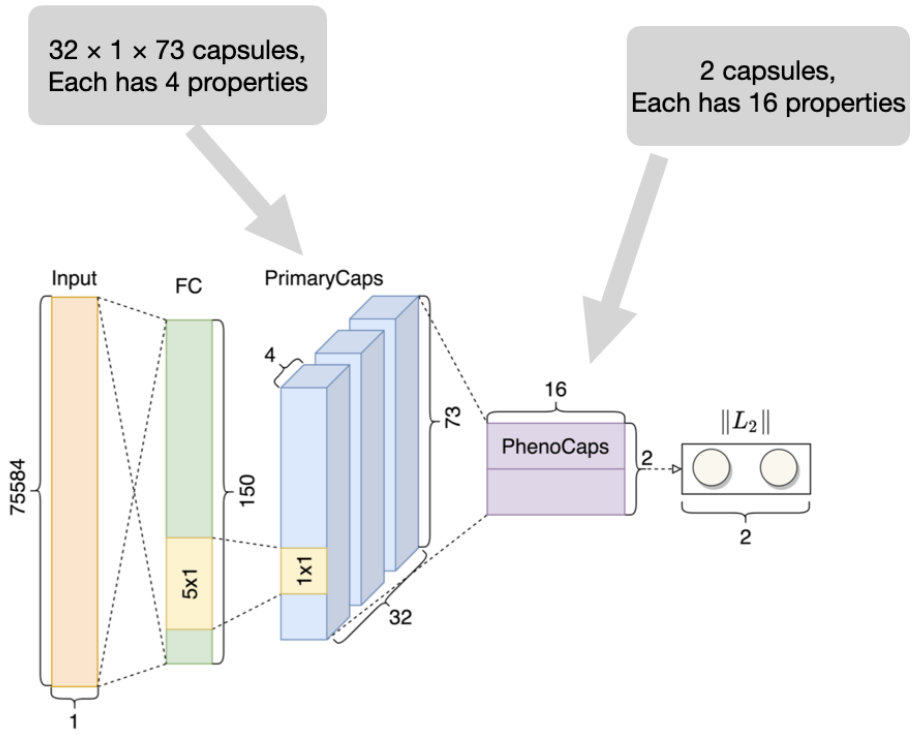
Gene-PCA

- Let $\mathcal{X}$ be all genotype profiles
- Let $\mathcal{X}(i) := \{X(i) \mid X \in \mathcal{X}\}$ be all partial genotype profiles, see (1)
- For each gene $g_i, i = 1, ..., M$, do
  1. Apply principal component analysis (PCA) to $\mathcal{X}(i)$
  2. Depending on $1 \leq N_i \leq 4$ (a), $4 < N_i \leq 20$ (b) or $N_i > 20$ (c), keep 1 (a), 4 (b) or 8 (c) principal components (PC's)
- This yielded 75584 PC's overall
- Representing each $X \in \mathcal{X}$ over the 75584 PC's turns

$$X \in \{0, 1, 2\}^N \quad \text{into} \quad \tilde{X} \in \mathbb{R}^{75584}$$

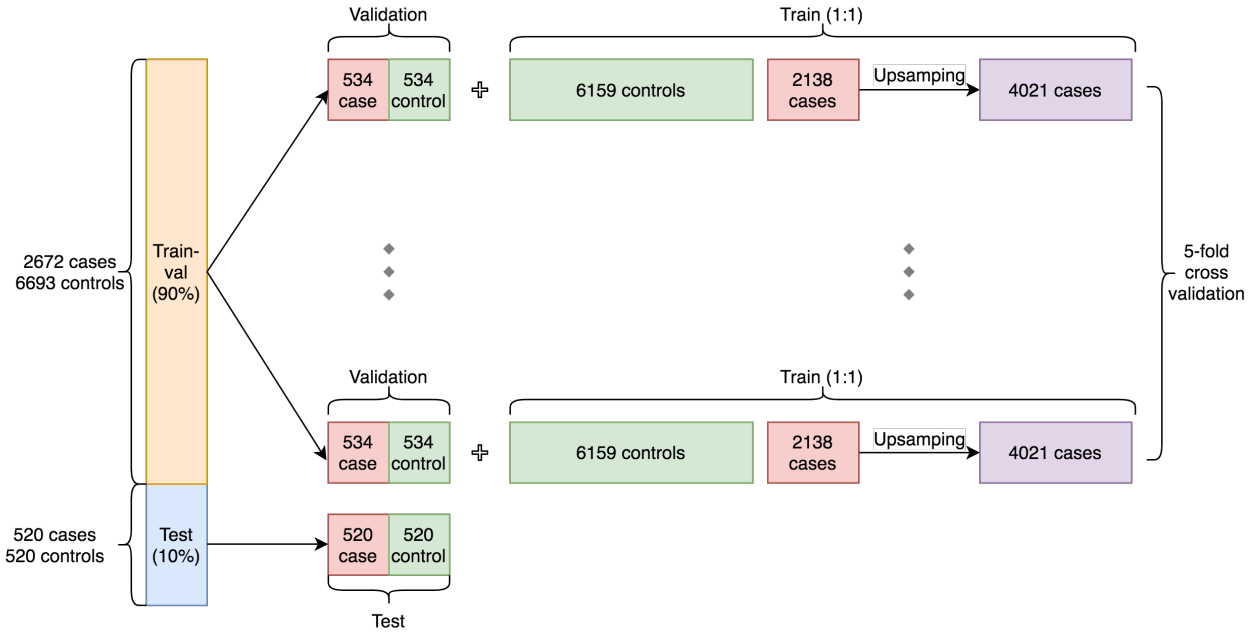
- Continue using $\tilde{X}$, instead of X as input

DiseaseCapsule: Network Architecture



- *Design:* Adopted from seminal work [Sabour et al, 2017]
- *Input:* Vector of length 75584 ☞ entries corresponding to Gene-PC's
- *Output:* Binary-valued, indicating disease or not

DiseaseCapsule: Training and Testing



- *Training:* 5-fold cross-validation
- *Testing:* Balanced test data set

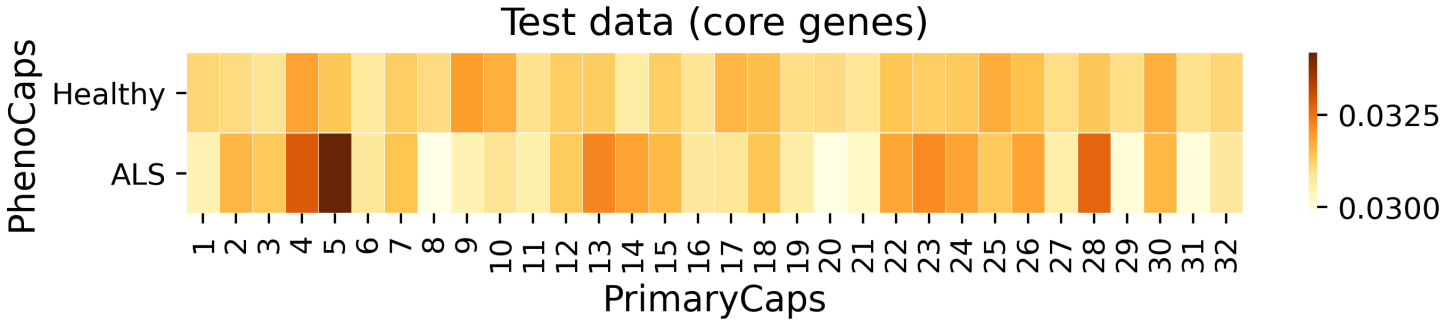
Results

Classification Performance

Dimension reduction	Classifier	Accuracy	Precision	Recall	F1-Score
Gene-PCA	DiseaseCapsule	**86.9**	85.2	89.4	**87.2**
Gene-PCA	MLP	84.2	92.2	74.8	82.6
Gene-PCA	Logistic Regression	78.2	71.1	94.8	81.3
Gene-PCA	SVM	76.3	**94.8**	55.8	70.3
Gene-PCA	CNN	74.5	86.1	58.5	69.7
Gene-PCA	Random Forest	63.3	73.0	42.1	53.4
Gene-PCA	AdaBoost	62.7	86.3	30.2	44.7
All-PCA	DiseaseCapsule	81.9	80.7	83.8	82.2
All-PCA	Logistic Regression	78.1	70.7	**96.0**	81.4
All-PCA	SVM	76.3	**94.8**	55.8	70.3
All-PCA	MLP	72.5	85.2	54.4	66.4
All-PCA	AdaBoost	67.6	84.8	42.9	57.0
All-PCA	Random Forest	64.1	73.3	44.4	55.3
All-PCA	CNN	53.8	54.8	42.5	47.9
-	Logistic Regression[a]	81.8	91.5	70.2	79.4
-	Logistic Regression[b]	78.5	84.4	69.8	76.4
-	Logistic Regression[c]	74.2	76.8	69.4	72.9
-	Logistic Regression[d]	63.5	63.5	63.3	63.4

Gene-PCA + DiseaseCapsule outperform alternative approaches

Interpretability – 922 Decisive Genes I

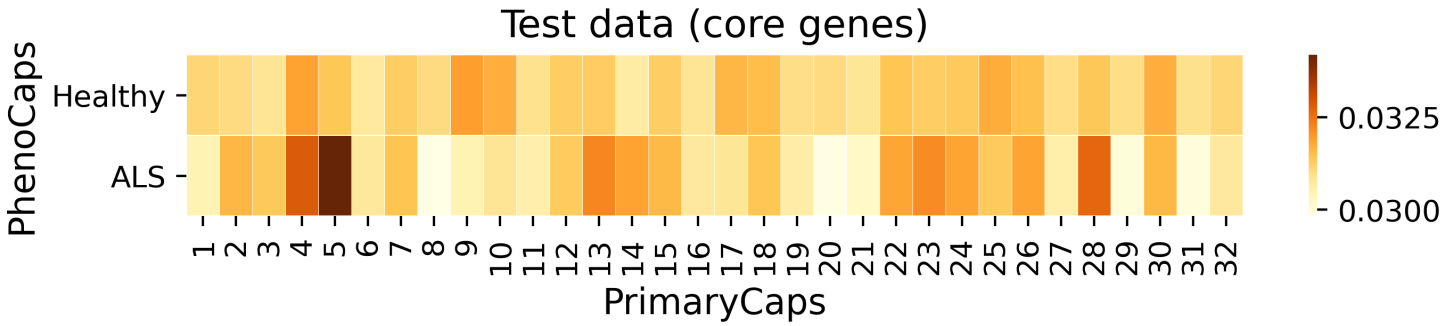


Strong link between primary capsule 5 and ALS pheno capsule

Selecting Genes that Drive Accurate Prediction: Protocol

- Primary capsule 5 activated by gene ensemble decisive for calling 'ALS'
- Select 922 genes that activate primary capsule 5 the most
- Annotating 922 genes revealed various reasonable, associated GO terms

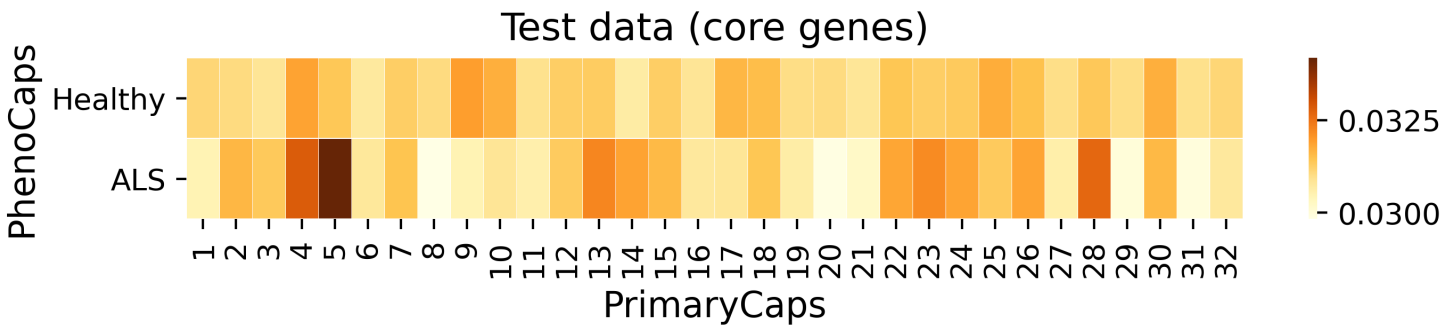
Interpretability – 922 Decisive Genes I



Strong link between primary capsule 5 and ALS pheno capsule

- Primary capsule 5 activated by gene ensemble decisive for calling 'ALS'
- Select 922 genes that activate primary capsule 5 the most
- Annotating 922 genes revealed various reasonable, associated GO terms

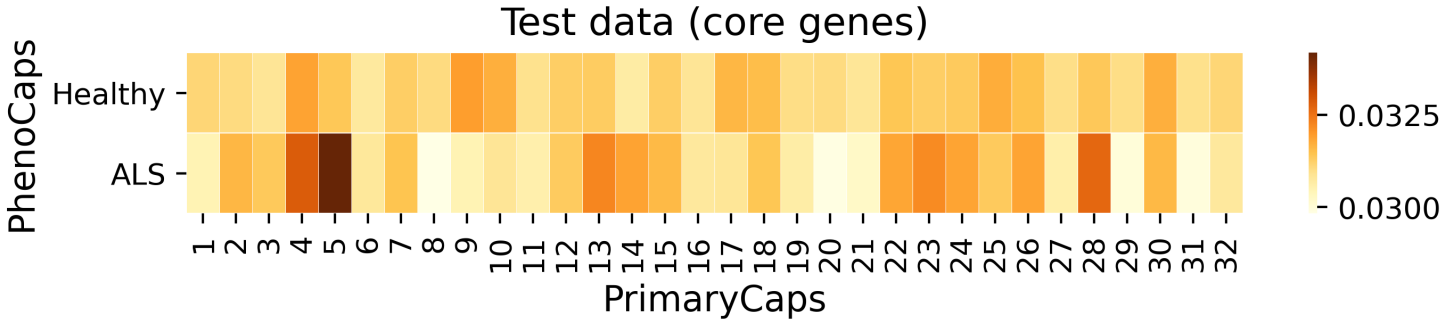
Interpretability – 922 Decisive Genes II



Strong link between primary capsule 5 and ALS pheno capsule

- Compute $c_{ij}, i = 1, ..., 32, j \in \{\text{Healthy}, \text{ALS}\}$ for each test sample
  - Remember that c_{ij} are determined individually for each sample
- Average c_{ij} obtained across all (here: 1040) test samples $\mathcal{X}_{\text{test}}$
- *Figure:* Averages for each combination i and j
- *Result:* c_{ij} is greatest for $i = 5, j = \text{ALS}$

Interpretability – 922 Decisive Genes III



Strong link between primary capsule 5 and ALS pheno capsule

Selecting Genes that Drive Accurate Prediction: Protocol

- Primary capsule 5 activated by gene ensemble decisive for calling 'ALS'
- Select 922 genes that activate primary capsule 5 the most
- Annotating 922 genes revealed various reasonable, associated GO terms

Interpretability – 922 Decisive Genes IV

- Consider a sample $\mathbf{x} \in \mathbb{R}^{75584}$ and one gene g
- Let $\mathbf{x}[n_g + 1], ..., \mathbf{x}[n_g + j_g]$ refer to the PC's of g
  - $n_g + 1$ indicates the position of the first PC of g within $\mathbf{x}$
  - *Recall:* $j_g \in \{1, 4, 8\}$, depending on number of variants in g
- Let $\mathbf{x}(g) \in \mathbb{R}^{75584}$ be defined by

$$\mathbf{x}(g) = [0, ..., 0, \mathbf{x}[n_g + 1], ..., \mathbf{x}[n_g + j_g], 0, ..., 0] \tag{2}$$

That is, all but the entries referring to g in $\mathbf{x}$ are turned to zero

Interpretability – 922 Decisive Genes V

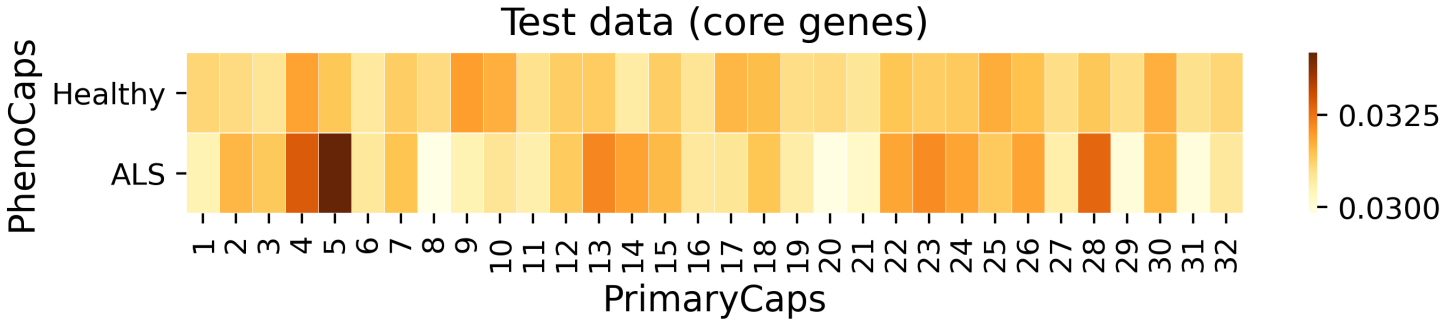
- Run trained Disease Capsule on all $\mathbf{x}(g)$ across all $\mathbf{x}$ and g
- Yields $c_{ij}^{\mathbf{x}(g)}$ for all combinations of $i, j, g, \mathbf{x}$
- For each gene g, average resulting $c_{5,\mathrm{ALS}}^{\mathbf{x}(g)}$ across all $\mathbf{x}$

$$c_{5,\mathrm{ALS}}^{g} = \frac{1}{|\mathcal{X}_{\mathrm{test}}| = 1040} \sum_{\mathbf{x} \in \mathcal{X}_{\mathrm{test}}} c_{5,\mathrm{ALS}}^{\mathbf{x}(g)} \tag{3}$$

  yielding gene specific coupling coefficients $c_{5,\mathrm{ALS}}^{g}$ for each g
- Keep the top 5% genes that yield the largest $c_{5,\mathrm{ALS}}^{g}$

☞ *Result:* 922 genes

Interpretability – 922 Decisive Genes VI



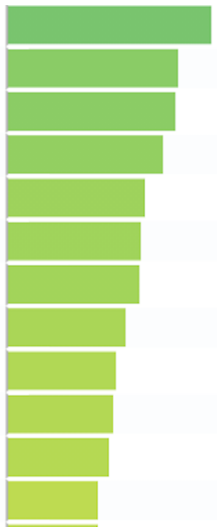
Strong link between primary capsule 5 and ALS pheno capsule

Selecting Genes that Drive Accurate Prediction: Protocol

- Primary capsule 5 activated by gene ensemble decisive for calling 'ALS'
- Select 922 genes that activate primary capsule 5 the most
- Annotating 922 genes revealed reasonable gene ontology (GO) terms

Interpretability – 922 Decisive Genes VII

nervous system development	GO:0007399	2.454×10^{-13}
neurogenesis	GO:0022008	2.655×10^{-11}
neuron differentiation	GO:0030182	3.942×10^{-11}
generation of neurons	GO:0048699	2.367×10^{-10}
neuron projection development	GO:0031175	2.801×10^{-9}
neuron development	GO:0048666	5.736×10^{-9}
anatomical structure development	GO:0048856	6.961×10^{-9}
system development	GO:0048731	4.750×10^{-8}
multicellular organism development	GO:0007275	1.937×10^{-7}
developmental process	GO:0032502	3.128×10^{-7}
central nervous system development	GO:0007417	5.386×10^{-7}
cell morphogenesis	GO:0000902	2.398×10^{-6}



First 12 GO Terms Associated with 922 Decisive Genes

1st / 2nd column: GO term / GO identifier
3rd, 4th column: significance of association

- Many annotations related with nervous system
- **Conclusion:** *Extraction of biomedical meaning from network possible*

Discovering 644 "Non-Additive" Genes

- Let G be the set of all 18 279 genes
- Let $S \subset G$ denote a subset of genes
- Let $\mathrm{ACC}_{\mathrm{DC}}(S)$ be the training accuracy achieved by Gene-PCA + DiseaseCapsule (DC) on genes S
- Let $\mathrm{ACC}_{\mathrm{LR}}(S)$ be the training accuracy of Gene-PCA + LogisticRegression (LR) on genes S
- *Goal:* Determine

$$\arg\max_{S \subset G} \mathrm{ACC}_{\mathrm{DC}}(S) - \mathrm{ACC}_{\mathrm{LR}}(S) \tag{4}$$

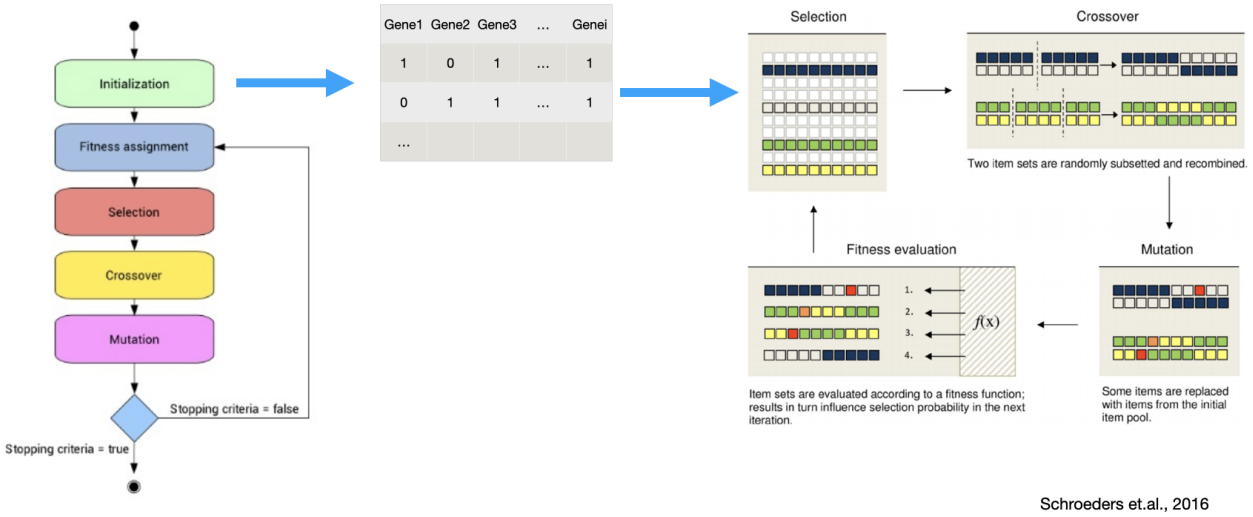
Discovering 644 "Non-Additive" Genes

- *Goal:* Determine

$$\arg\max_{S \subset G} \mathrm{ACC}_{\mathrm{DC}}(S) - \mathrm{ACC}_{\mathrm{LR}}(S)$$

- The resulting S yields the greatest difference in performance between DC and LR
- *Insight:* This S must bring up non-additive effects to do this
- *Question:* How to determine the optimal S?

Discovering 644 "Non-Additive" Genes



Genetic Algorithm: Workflow

- *Idea:* Use genetic algorithm for determining S that maximizes $\mathrm{ACC}_{\mathrm{DC}}(S) - \mathrm{ACC}_{\mathrm{LR}}(S)$
- *Result:* 644 genes yield maximal gain of non-linear DC over linear LR

Discovering 644 "Non-Additive" Genes II

Model	Accuracy	precison	Recall
LogisticReg	0.512	0.522	0.292
DiseaseCapsule	0.712	0.715	0.706
Difference	0.200	0.193	0.414

Classification Performance on 644 "Non-Additive" Genes

- *Experiment:* Retrain DiseaseCapsule and Logistic Regression using only 644 "non-additive" genes
- *Result:*
  - 644 "non-additive" genes do not work in linear regression scheme
  - 644 "non-additive" genes work excellently in DiseaseCapsule

DiseaseCapsule Needs Less Training Data

DR **Classifiers**	Gene-PCA DiseaseCapsule	Gene-PCA MLP	Gene-PCA LR	Gene-PCA SVM	Gene-PCA CNN	- PRS*
5%	**74.0** $\pm$ 1.4	61.5 $\pm$ 1.1	69.1 $\pm$ 0.5	50.0 $\pm$ 0.0	53.7 $\pm$ 2.0	60.1 $\pm$ 1.1
10%	**79.0** $\pm$ 0.8	66.0 $\pm$ 0.9	72.1 $\pm$ 0.5	53.2 $\pm$ 0.3	56.2 $\pm$ 1.6	67.0 $\pm$ 0.8
20%	**81.3** $\pm$ 0.6	71.4 $\pm$ 0.6	73.4 $\pm$ 0.3	61.4 $\pm$ 0.2	61.7 $\pm$ 2.0	71.9 $\pm$ 0.8
40%	**83.5** $\pm$ 0.5	76.5 $\pm$ 0.6	74.6 $\pm$ 0.4	67.3 $\pm$ 0.3	66.8 $\pm$ 0.9	76.9 $\pm$ 0.9
60%	**84.2** $\pm$ 0.3	79.9 $\pm$ 0.7	76.1 $\pm$ 0.3	71.1 $\pm$ 0.1	70.2 $\pm$ 1.3	79.3 $\pm$ 0.6
80%	**85.1** $\pm$ 0.4	81.5 $\pm$ 0.6	76.7 $\pm$ 0.3	73.2 $\pm$ 0.2	72.3 $\pm$ 1.1	80.8 $\pm$ 0.5
100%	**86.3** $\pm$ 0.3	83.6 $\pm$ 0.9	78.6 $\pm$ 0.3	76.1 $\pm$ 0.2	73.7 $\pm$ 0.7	81.9 $\pm$ 0.3

Classification Performance Relative to Size of Training Data

- DC's performance stable on decreasing training data
- Other methods: performance collapses
- Only *exception:* logistic regression (LR)

DiseaseCapsule: Summary

- DiseaseCapsule shows superior accuracy in prediction
- DiseaseCapsule opens up interesting ways for interpreting results
  - DiseaseCapsule reveals genes that are decisive for classification
  - DiseaseCapsule reveals genes that do not add up their effects
- DiseaseCapsule requires less training data

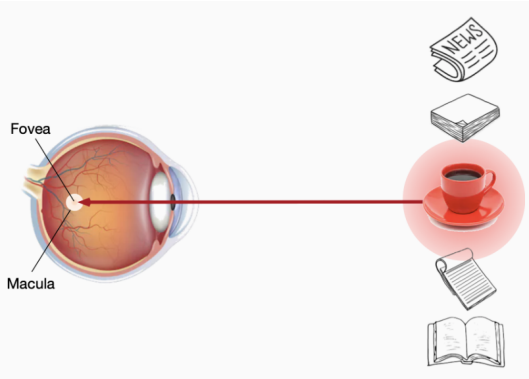
Attention

Attention: Biological Motivation

Attention: Motivation I

- Optic nerve receives 10^8 bits per second
- *Challenge:* Distinguish between important and irrelevant information
- *Solution: Attention*
  - Brain focuses on only a fraction of information
  - Smart usage of resources
  - Brain needs to know where to direct attention
- *Idea:* William James, "father of American psychology", 1890's
- Distinguish between *non-volitional* and *volitional cues*
  - They trigger subconscious and conscious actions

Attention: Nonvolitional Cues

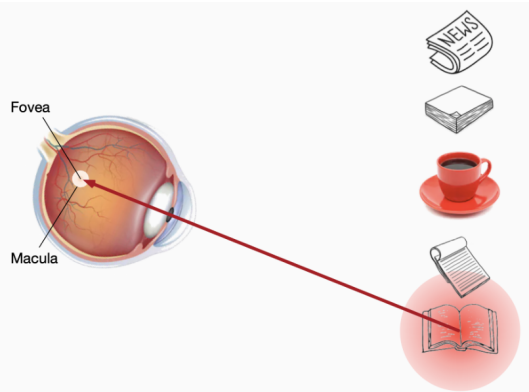


Nonvolitional cue: eye directs attention *non-voluntarily* to red coffee cup

From `https://d2l.ai`

- Nonvolitional cues based on saliency / conspicuity of objects
- *Example:*
  - Papers on desk black and white
  - Coffee cup red
  - *Consequence:* Eye "sees" coffee cup first
    ☞ Person grabs and drinks coffee

Attention: Volitional Cues



Deliberately searching for entertainment, eye *voluntarily* directs attention to book

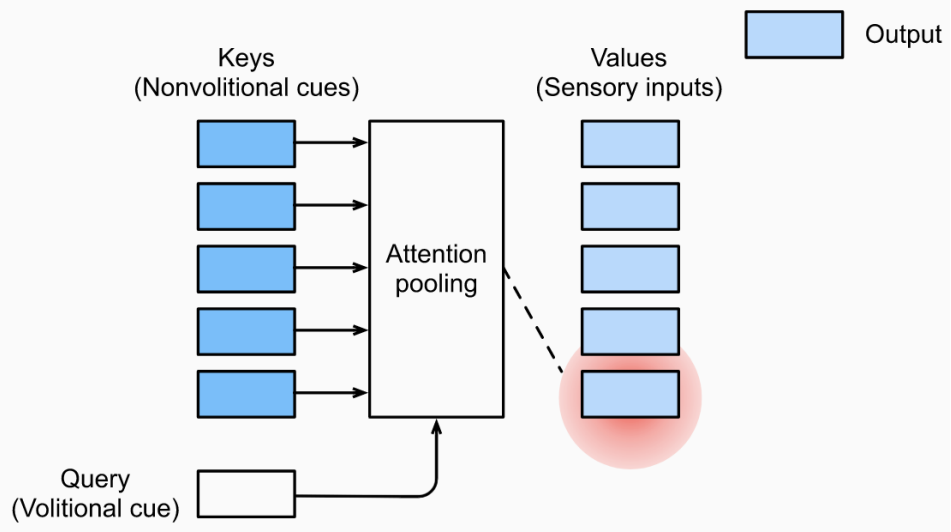
From `https://d2l.ai`

- Done with coffee, brain wants entertainment
- *Consequence:* Eye "sees" book in a deliberate attempt
- *Task-oriented search:*
  - Brain pre-trained to recognize objects that promise entertainment
  - Selection of book under full cognitive and volitional control

Queries, Keys and Values

Attention: Queries, Keys and Values I

Motivation

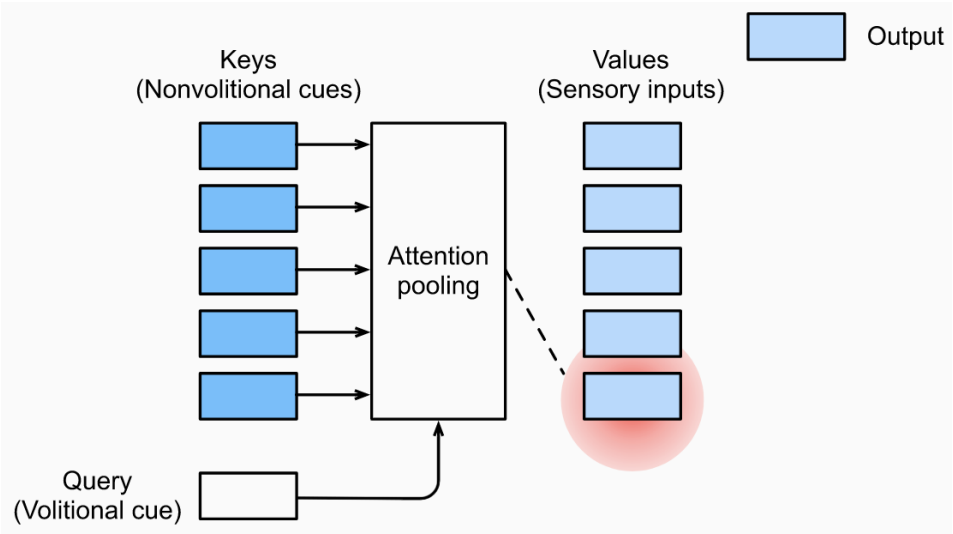


Attention pooling: integrating queries with keys (input) and values (output)

- Ordinary networks reflect non-volitional attention
- *Examples:* Convolutional and fully connected networks
- *Goal:* Model volitional attention cues and integrate them appropriately

Attention: Queries, Keys and Values II

Solution

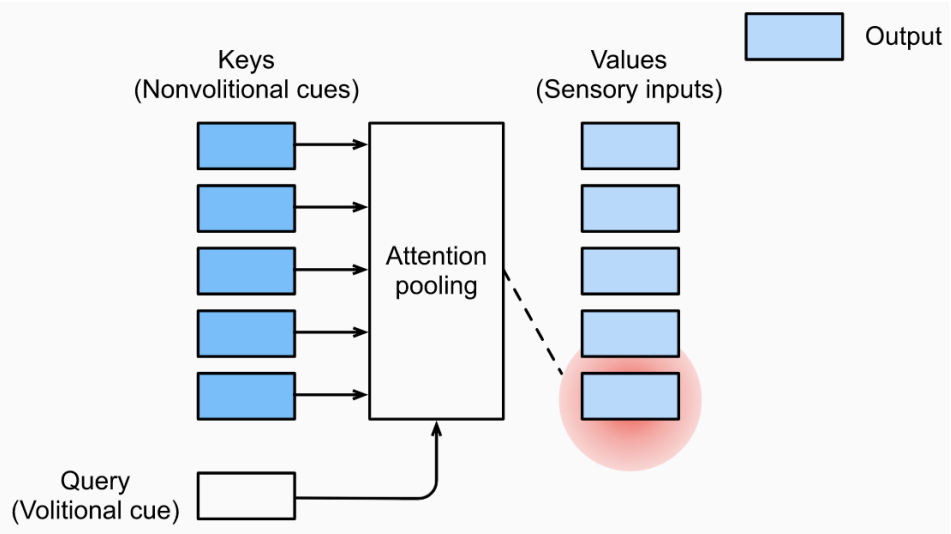


Attention pooling: integrating queries with keys (input) and values (output)

- Input / output ordinary neurons: *keys* and *values*
- Keys and values come in pairs
- Volitional cues = *queries*
- Model patterned after searches in databases

Attention: Queries, Keys and Values III

Attention Pooling

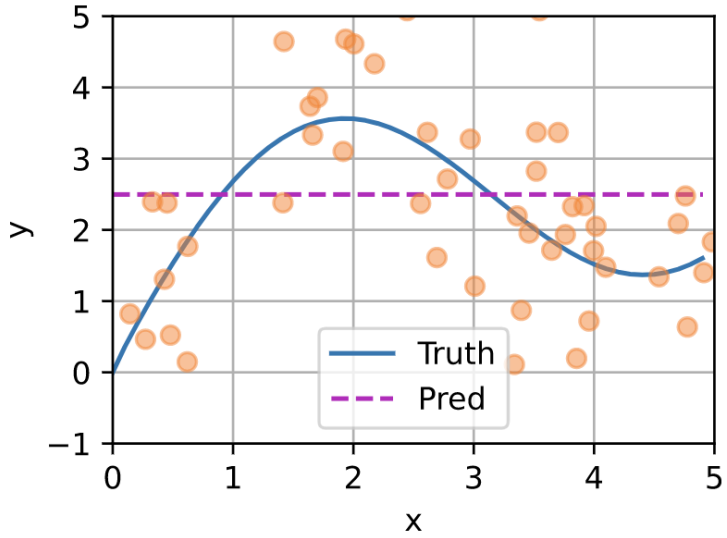


Attention pooling: integrating queries with keys (input) and values (output)

- Computes *attention weights* for each key
- Attention weight reflects compatibility of key and query
- Attention pooling computes weighted sum of values
- *Output* dominated by value whose key matches query best

Attention Pooling

Attention Average Pooling



From `https://d2l.ai`

- *Truth:* $y = f(x) := 2\sin(x) + x^{0.8}$ (blue)
- *Data points* (x_i, y_i) sampled from $y_i = f(x_i) + \epsilon$ where ϵ follows normal distribution with $\mu = 0, \sigma = 0.5$ (orange dots)
- *Prediction:* $\hat{f}(x) := \sum_{i=1}^{n} y_i$ where $n = \#$ training data (dashed pink)
  - Reflects unweighted average pooling
- *Conclusion:* Unweighted average pooling not necessarily good idea

Nadaraya-Watson Kernel Regression I

- Let $K(.)$ be a *kernel*
- *Kernel properties:*
  - $K(x) \to 0$ for $|x| \to \infty$
  - $K(0)$ is maximum
- *Example: Gaussian kernel*

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{u^2}{2}) \tag{5}$$

- *Nadaraya-Watson kernel regression:* For unseen x, determine

$$\hat{f}(x) = \sum_{i=1}^{n} \frac{K(x - x_i)}{\sum_{j=1}^{n} K(x - x_j)} y_i \tag{6}$$

where $(x_i, y_i), i = 1, ..., n$ are the training data points

Nadaraya-Watson Kernel Regression II

- *Nadaraya-Watson kernel regression:* For unseen x, determine

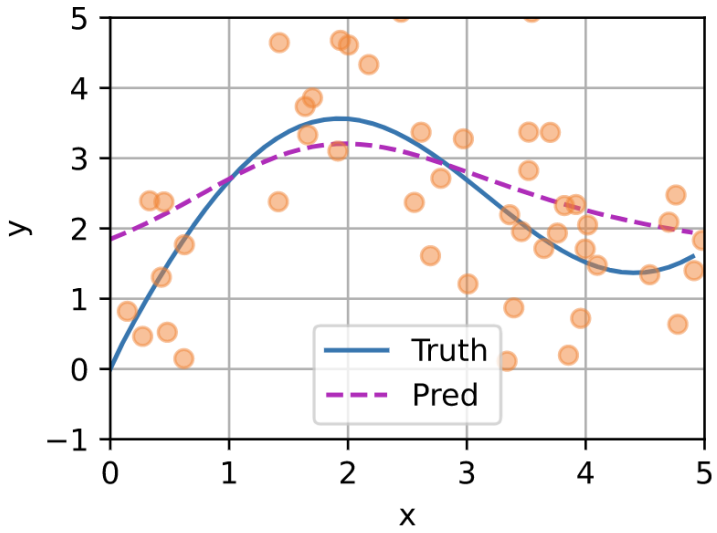
$$\hat{f}(x) = \sum_{i=1}^{n} \frac{K(x - x_i)}{\sum_{j=1}^{n} K(x - x_j)} y_i \tag{7}$$

  where $(x_i, y_i), i = 1, ..., n$ are the training data points
- This agrees with general concept of attention pooling

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha(x, x_i) y_i \tag{8}$$

  where x is query, and (x_i, y_i) are key-value pairs
- Value y_i receives more weight the closer its key x_i to x

Nadaraya-Watson Kernel Regression III

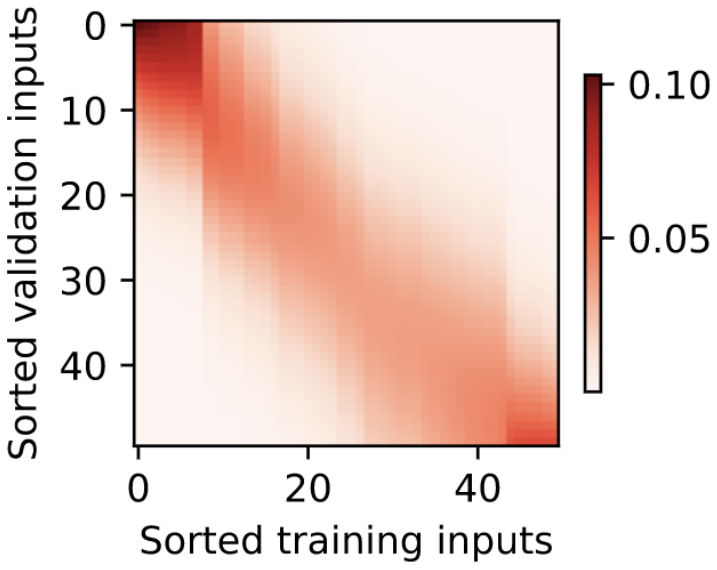


From `https://d2l.ai`

- Plugging the Gaussian kernel (5) into (7),(8) yields (dashed pink curve)

$$\begin{aligned}\hat{f}(x) &= \sum_{i=1}^{n} \alpha(x, x_i) y_i = \sum_{i=1}^{n} \frac{\exp(-\frac{1}{2}(x - x_i)^2)}{\sum_{j=1}^{n} \exp(-\frac{1}{2}(x - x_j)^2)} y_i \\ &= \sum_{i=1}^{n} \mathrm{softmax}(-\frac{1}{2}(x - x_i)^2) y_i\end{aligned} \tag{9}$$

Nadaraya-Watson Kernel Regression IV



From `https://d2l.ai`

- 50 training data points (x_i, y_i)
- 50 validation data points x
- Sort training and validation data by x_i and x resp.
- Plot $\alpha(x, x_i) = \sum_{i=1}^{n} \mathrm{softmax}(-\frac{1}{2}(x - x_i)^2)$ for each pair (x_i, x)

Nadaraya-Watson Kernel Regression V

- Nadaraya-Watson kernel regression

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha(x, x_i) y_i = \sum_{i=1}^{n} \frac{K(x - x_i)}{\sum_{j=1}^{n} K(x - x_j)} y_i \tag{10}$$

  is an example of *nonparametric attention pooling*
- *Benefit:* Converges to true function on increasing training data
  - *Reminder:* Training data reflect key-value pairs
- *Disadvantage:* There are no learnable parameters

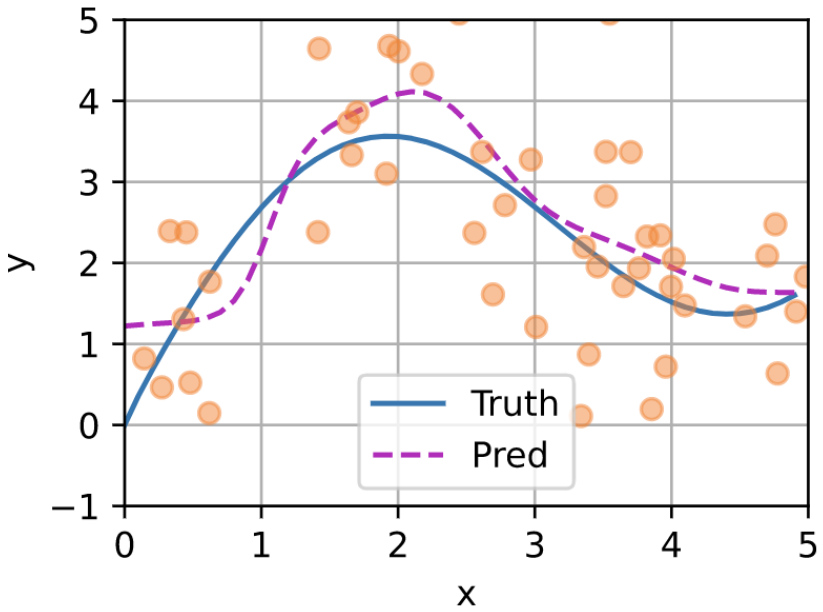
Parametric Attention Pooling I

- Integration of a learnable parameter w into (9) yields

$$\begin{aligned}\hat{f}(x) &= \sum_{i=1}^{n} \alpha(x, x_i) y_i = \sum_{i=1}^{n} \frac{\exp(-\frac{1}{2}((x - x_i)w)^2)}{\sum_{j=1}^{n} \exp(-\frac{1}{2}((x - x_j)w)^2)} y_i \\ &= \sum_{i=1}^{n} \mathrm{softmax}(-\frac{1}{2}((x - x_i)w)^2) y_i\end{aligned} \tag{11}$$

- The parameter w can be learnt via (stochastic) gradient descent
- w reflects influence span of keys on queries
  - Number of influential keys decreases on increasing w

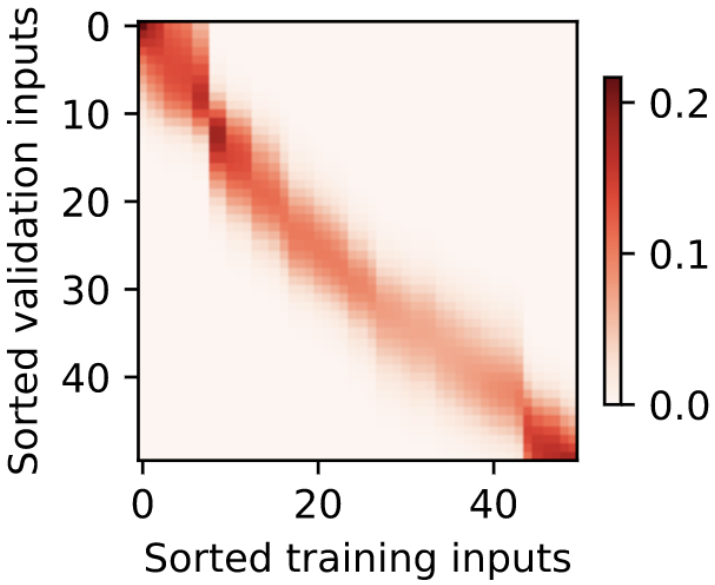
Parametric Attention Pooling II



From `https://d2l.ai`

- Predicted curve is less smooth than nonparametric counterpart

Parametric Attention Pooling III



From `https://d2l.ai`

- Training / validation procedure analogous to nonparametric setting
- However, training includes learning parameter w
- Region with larger attention weights sharper in parametric setting

Attention Scoring Functions

Attention Pooling: Digest I

- Re-consider (9):

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha(x, x_i) y_i = \sum_{i=1}^{n} \text{softmax}(-\frac{1}{2}(x - x_i)^2) y_i$$

- One can view $\alpha(x, x_i)$ as
  - an attention scoring function

$$a(x, x_i) := -\frac{1}{2}(x - x_i)^2 \tag{12}$$

  - that is further fed into a softmax operation, yielding

$$\alpha(x, x_i) = \text{softmax}(a(x, x_i)) \tag{13}$$

Attention Pooling: Digest II

- One can view $\alpha(x, x_i)$ as
  - an attention scoring function

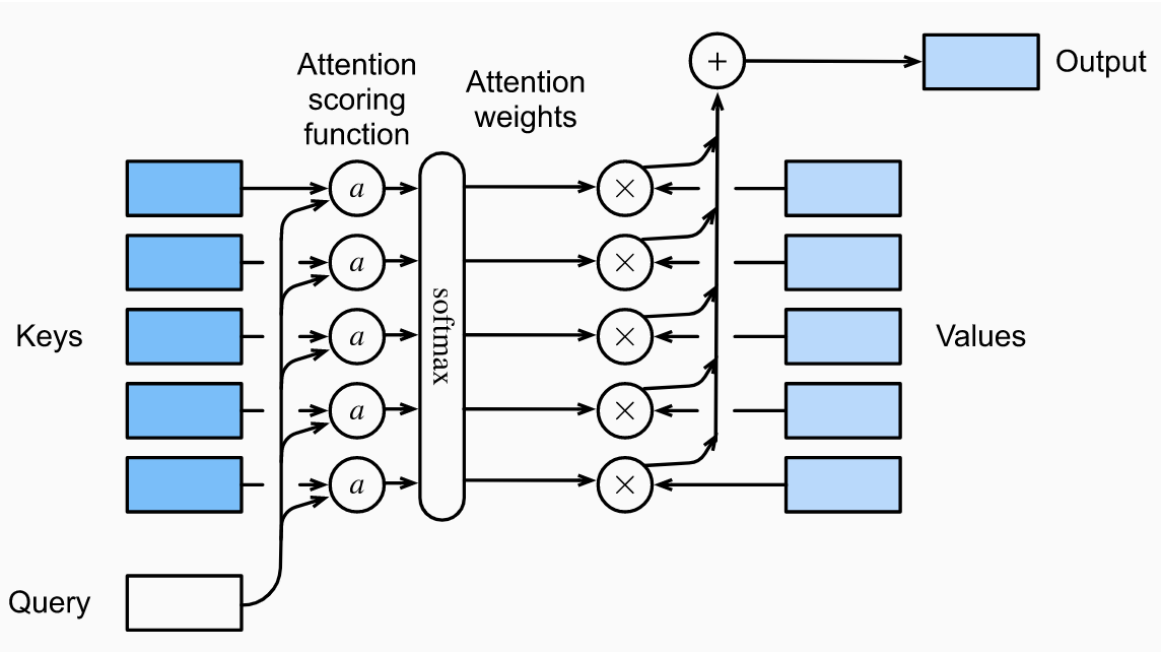
$$a(x, x_i) := -\frac{1}{2}(x - x_i)^2 \tag{14}$$

  - that is further fed into a softmax operation, yielding

$$\alpha(x, x_i) = \text{softmax}(a(x, x_i)) \tag{15}$$

- *Result:* Probability distribution
  - over values y_i paired with keys x_i where
  - probabilities are attention weights $\alpha(x, x_i)$

Attention Scoring Functions: Motivation



Output of attention pooling is weighted average of values

- Let x be query, and x_i keys. Attention weights generally compute as

$$\alpha(x, x_i) = \text{softmax}(a(x, x_i)) \tag{16}$$

- *Advantage:* Freedom in choosing attention scoring functions $a(x, x_i)$

Attention Pooling: Formal Summary

- Let $\mathbf{q} \in \mathbb{R}^q$ be a query and $(\mathbf{k}_1, \mathbf{v}_1), ..., (\mathbf{k}_m, \mathbf{v}_m), \mathbf{k}_i \in \mathbb{R}^k, \mathbf{v}_i \in \mathbb{R}^v$ be m key-value pairs
- The *attention pooling* f computes as

$$f(\mathbf{q}, (\mathbf{k}_1, \mathbf{v}_1), ..., (\mathbf{k}_m, \mathbf{v}_m)) = \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i \in \mathbb{R}^v \tag{17}$$

- The *attention weight* $\alpha(\mathbf{q}, \mathbf{k}_i) \in \mathbb{R}$ computes as

$$\alpha(\mathbf{q}, \mathbf{k}_i) = \text{softmax}(a(\mathbf{q}, \mathbf{k}_i)) = \frac{\exp(a(\mathbf{q}, \mathbf{k}_i)}{\sum_{j=1}^{m} \exp(a(\mathbf{q}, \mathbf{k}_j))} \tag{18}$$

- The *attention scoring function* $a(\mathbf{q}, \mathbf{k})$ maps two vectors to a scalar

$$a : \mathbb{R}^q \times \mathbb{R}^k \longrightarrow \mathbb{R} \tag{19}$$

Additive Attention Scoring

- Let $\mathbf{q} \in \mathbb{R}^q$ be a query and $\mathbf{k} \in \mathbb{R}^k$ be a key
- Let $\mathbf{W}_q \in \mathbb{R}^{h \times q}, \mathbf{W}_k \in \mathbb{R}^{h \times k}, \mathbf{w}_v \in \mathbb{R}^h$ collect learnable parameters
- The *additive attention scoring function* computes as

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^T \tanh(\mathbf{W}_q \mathbf{q} + \mathbf{W}_k \mathbf{k}) \in \mathbb{R} \tag{20}$$

- *Interpretation:* (20) reflects running $\mathbf{q}, \mathbf{k}$ through MLP
  - *Input:* Concatenation of $\mathbf{q}$ and $\mathbf{k}$
  - One *hidden layer* of width h
  - Parameters from input to hidden layer are $\mathbf{W}_q, \mathbf{W}_k$
  - The activation function is tanh
  - Parameters from hidden to output layer captured by $\mathbf{w}_v$

Scaled Dot-Product Attention Scoring

- Let $\mathbf{q}, \mathbf{k} \in \mathbb{R}^d$ be *equal-sized* query and key
- The *scaled dot-product attention scoring function* computes as

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T \mathbf{k} / \sqrt{d} \tag{21}$$

- *Note:* Dot product $\mathbf{q}^T \mathbf{k}$ has mean 0 and variance d
  ☞ Dividing by $\sqrt{d}$ implies standard deviation of 1

Minibatches:

- Computing attention for n queries and m keys at once
- For queries $\mathbf{Q} \in \mathbb{R}^{n \times d}$, keys $\mathbf{K} \in \mathbb{R}^{m \times d}$, values $\mathbf{V} \in \mathbb{R}^{m \times v}$ compute

$$\text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V} \in \mathbb{R}^{n \times v} \tag{22}$$

Multi-Head Attention

Multi-Head Attention I

- *Motivation:* Capture different attention mechanisms for same queries, keys, values
- *Example:* Attend to both short- and long-range dependencies in sequential data
- *Question:* How to vary attention mechanisms in informed way?

Multi-Head Attention II

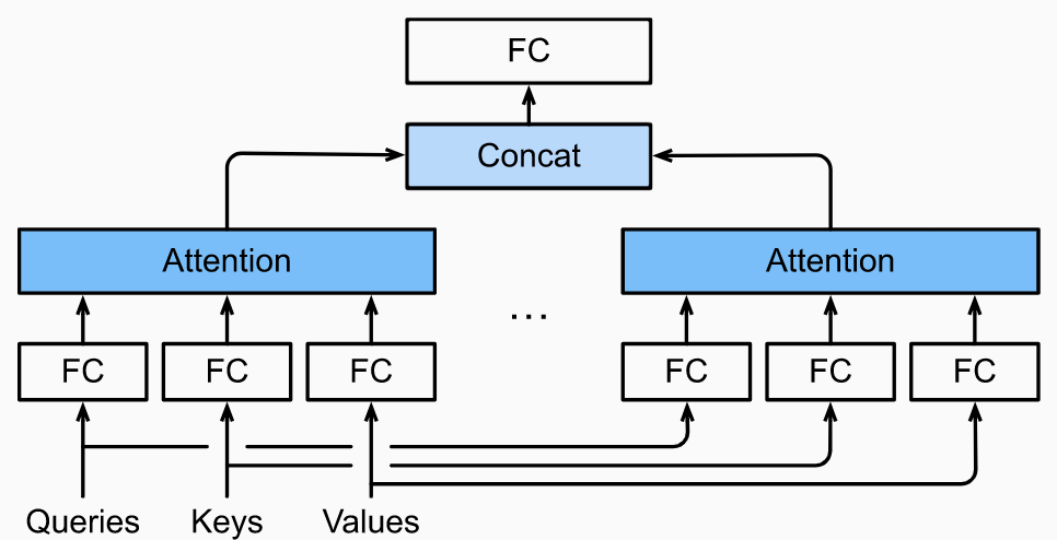
- *Question:* How to vary attention mechanisms in informed way?
- *Solution:*
  - Let h be intended number of attention mechanisms
  - Linearly transform queries, keys, values using h different sets of matrices $\mathbf{W}_i^{(q)}, \mathbf{W}_i^{(k)}, \mathbf{W}_i^{(v)}, i = 1, \ldots, h$
  - Run the h differently transformed queries, keys, values through attention pooling
  - Transformations $\mathbf{W}_i^{(q)}, \mathbf{W}_i^{(k)}, \mathbf{W}_i^{(v)}, i = 1, \ldots, h$ are learnt
  - The h attention pooling outputs are concatenated, and linearly transformed by another learned matrix $\mathbf{W}_o$
- Design is called *multi-head attention*
- Each of the h attention pooling outputs is referred to as a *head*

Multi-Head Attention III

- Let $\mathbf{q} \in \mathbb{R}^{d_q}, \mathbf{k} \in \mathbb{R}^{d_k}, \mathbf{v} \in \mathbb{R}^{d_v}$ be query, key, value
- Let $\mathbf{W}_i^{(q)} \in \mathbb{R}^{p_q \times d_q}, \mathbf{W}_i^{(k)} \in \mathbb{R}^{p_k \times d_k}, \mathbf{W}_i^{(v)} \in \mathbb{R}^{p_v \times d_v}$ collect learnable parameters
- f is attention pooling, such as additive (20) or dot-product attention (21)
- Each attention head is computed as

$$\mathbf{h}_i = f(\mathbf{W}_i^{(q)}\mathbf{q}, \mathbf{W}_i^{(k)}\mathbf{k}, \mathbf{W}_i^{(v)}\mathbf{v}) \in \mathbb{R}^{p_v} \tag{23}$$

Multi-Head Attention IV



From `https://d2l.ai`

- Attention heads:

$$\mathbf{h}_i = f(\mathbf{W}_i^{(q)}\mathbf{q}, \mathbf{W}_i^{(k)}\mathbf{k}, \mathbf{W}_i^{(v)}\mathbf{v}) \in \mathbb{R}^{p_v} \tag{24}$$

- Initial 'FC' layers reflect operations $\mathbf{W}_i^{(q)}\mathbf{q}, \mathbf{W}_i^{(k)}\mathbf{k}, \mathbf{W}_i^{(v)}\mathbf{v}$
- 'Attention' layers reflect application of f to $\mathbf{W}_i^{(q)}\mathbf{q}, \mathbf{W}_i^{(k)}\mathbf{k}, \mathbf{W}_i^{(v)}\mathbf{v}$

Multi-Head Attention V

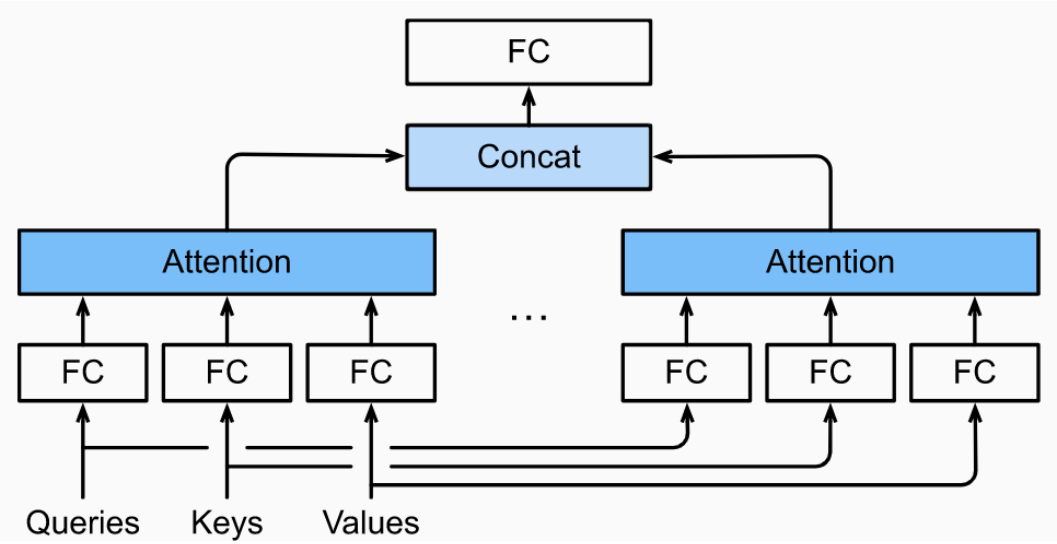
- Attention heads:

$$\mathbf{h}_i = f(\mathbf{W}_i^{(q)}\mathbf{q}, \mathbf{W}_i^{(k)}\mathbf{k}, \mathbf{W}_i^{(v)}\mathbf{v}) \in \mathbb{R}^{p_v} \tag{25}$$

- Let $\mathbf{W}_o \in \mathbb{R}^{p_o \times hp_v}$ collect further learnable parameters
- The multi-head attention output computes as

$$\mathbf{W}_o \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_h \end{bmatrix} \in \mathbb{R}^{p_o} \tag{26}$$

Multi-Head Attention II



From `https://d2l.ai`

- Multi-head attention output computes as

$$\mathbf{W}_o[\mathbf{h}_1^T, \ldots, \mathbf{h}_h^T]^T \in \mathbb{R}^{p_o} \tag{27}$$

- 'Concat' layer reflects forming $[\mathbf{h}_1^T, \ldots, \mathbf{h}_h^T]$
- Final 'FC' layer reflects application of $\mathbf{W}_o$

References

- `http://d2l.ai`, 10.6, 10.7, 11.1–11.3, 11.5

Outlook

- Sequence-2-Sequence Models
- Attention Mechanisms II
- Biological Applications

Thanks for your attention