Lecture Notes for

Neural Networks and Machine Learning

Paired Losses, Multi-task, and Multi-Modal Learning

Logistics and Agenda

- Logistics
  - Grading update
- Agenda
  - Consistency, Contrastive, and Triplet Loss
  - Multi-modal and Multi-Task
- Next Time
  - Finish Multi-modal and Multi-Task
  - Demo

Last Time: Consistency loss

Neural Network approximates $p(y|x)$ by $\mathbf{w}$
Use labeled data to minimize network

Sample new $\mathbf{x}$ from unlabeled pool with function q
function q is augmentation procedure
Minimize cross entropy of two models

intuition of final model

$$\mathcal{D}_{KL}(f\,||\,g) = -\sum f(x)\cdot\log\frac{g(x)}{f(x)}$$ definition of Kullback-Leibler (KL) Divergence

$$\mathcal{D}_{KL}(p_{\mathbf{w}}(y\,|\,\mathbf{x})\,||\,p_{\mathbf{w}}(y\,|\,\hat{\mathbf{x}}))$$

$$\mathcal{D}_{KL}(p(y\,|\,\mathbf{x})\,||\,p(y\,|\,\hat{\mathbf{x}})) = -\sum p(y\,|\,\mathbf{x})\cdot\log\frac{p(y\,|\,\hat{\mathbf{x}})}{p(y\,|\,\mathbf{x})} = -\sum p(y\,|\,\mathbf{x})\cdot\left(\log p(y\,|\,\hat{\mathbf{x}}) - \log p(y\,|\,\mathbf{x})\right)$$

$$= -\sum p(y\,|\,\mathbf{x})\cdot\log p(y\,|\,\hat{\mathbf{x}}) + \sum \cancel{p(y\,|\,\mathbf{x})\cdot\log p(y\,|\,\mathbf{x})}$$

constant

cross entropy of augmented and not augmented model

global entropy of model output constant as $p(y\,|\,\mathbf{x})$ has no uncertainty

```
cce = tf.keras.losses.CategoricalCrossentropy()
cce(y_pred, y_pred_augmented)
```

mathematics with strict assumptions

		Semi-supervised setting					
Initialization	**UDA**	IMDb (20)	Yelp-2 (20)	Yelp-5 (2.5k)	Amazon-2 (20)	Amazon-5 (2.5k)	DBpedia (140)
Random	✗	43.27	40.25	50.80	45.39	55.70	41.14
	✓	25.23	8.33	41.35	16.16	44.19	7.24
$\text{BERT}_{\text{BASE}}$	✗	18.40	13.60	41.00	26.75	44.09	2.58
	✓	5.45	2.61	33.80	3.96	38.40	1.33
$\text{BERT}_{\text{LARGE}}$	✗	11.72	10.55	38.90	15.54	42.30	1.68
	✓	4.78	2.50	33.54	3.93	37.80	1.09
$\text{BERT}_{\text{FINETUNE}}$	✗	6.50	2.94	32.39	12.17	37.32	-
	✓	**4.20**	**2.05**	**32.08**	**3.50**	**37.12**	-

Error Rates

Contrastive Loss

Jürgen Schmidhuber
@SchmidhuberAI

DeepSeek [1] uses elements of the 2015 reinforcement learning prompt engineer [2] and its 2018 refinement [3] which collapses the RL machine and world model of [2] into a single net through the neural net distillation procedure of 1991 [4]: a distilled chain of thought system.

REFERENCES (easy to find on the web):

[1] #DeepSeekR1 (2025): Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv 2501.12948

[2] J. Schmidhuber (JS, 2015). On Learning to Think: Algorithmic Information Theory for Novel Combinations of Reinforcement Learning Controllers and Recurrent Neural World Models. arXiv 1210.0118. Sec. 5.3 describes the reinforcement learning (RL) prompt engineer which learns to actively and iteratively query its model for abstract reasoning and planning and decision making.

[3] JS (2018). One Big Net For Everything. arXiv 1802.08864. See also US11853886B2. This paper collapses the reinforcement learner and the world model of [2] (e.g., a foundation model) into a single network, using the neural network distillation procedure of 1991 [4]. Essentially what's now called an RL "Chain of Thought" system, where subsequent improvements are continually distilled into a single net. See also [5].

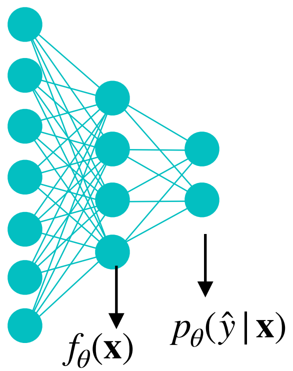
Dealing with Data Sparsity

- $$\mathcal{L}_{ce}(y, \hat{y}) = -\sum_{i \in L} p(y^{(i)} = c) \cdot \log\left(p_\theta(\hat{y}^{(i)} = c \,|\, \mathbf{x}^{(i)})\right)$$
- **Problem**: When we have a limited number of labeled samples, L, there are also a limited number of gradient updates
  - Can we boost gradient updates existing labeled data, perhaps even *exponentially*?
  - Can latent space distances be made meaningful?
- **Contrastive Loss:**
  - Use a metric to measure similarity of samples within latent space (e.g., cosine distance, euclidean, etc.),
  - Randomly sample from two or more classes
  - Push same classes together
  - Push different classes apart (within a threshold)

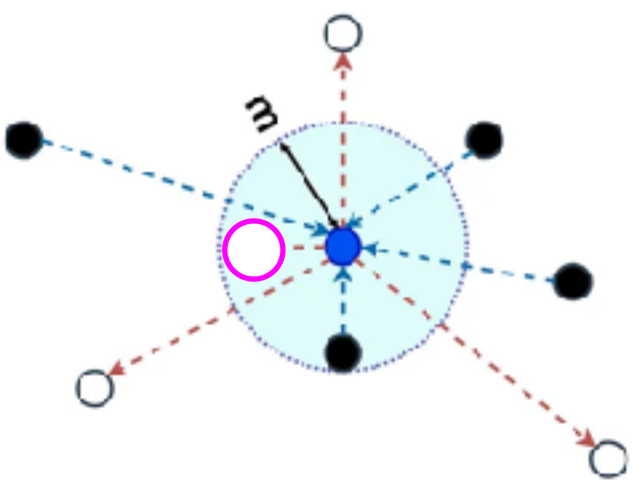
Contrastive loss

Latent representations of $\mathbf{x}$, from model f

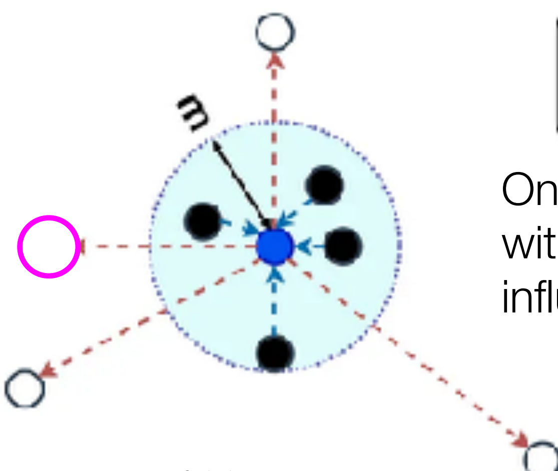
$$D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \| f_\theta(\mathbf{x}^{(i)}) - f_\theta(\mathbf{x}^{(j)}) \|_2$$



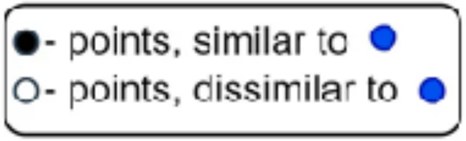
$$\mathcal{L}_c = \underbrace{\sum_{i,j \in S} D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})}_{\text{similar}} + \underbrace{\sum_{i,j \in \hat{S}} \max\left(0, m - D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})\right)}_{\text{not similar}}$$



$f_\theta(\mathbf{x})$ **Before**



$f_\theta(\mathbf{x})$ **After**



Only dissimilar points within m are influenced

https://medium.com/@maksym.bekuzarov/losses-explained-contrastive-loss-f8f57fe32246

Contrastive loss in Face Detect/Identify

Contrastive Loss great for authenticating without re-training

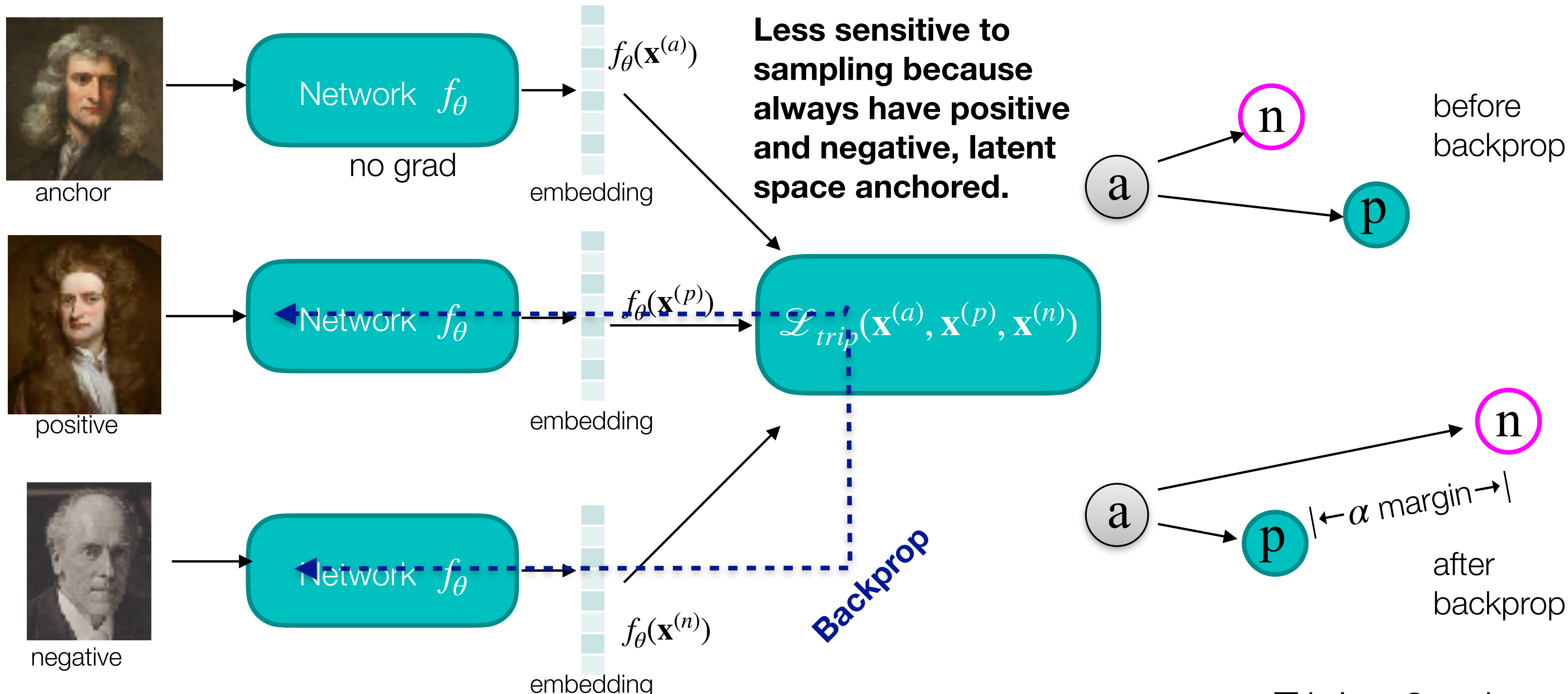
However, the **training pairs chosen for positive and negative samples** tends to be **sensitive to sampling** for good performance.

Traditional ML

Contrastive Model

Triplet Loss, $\mathscr{L}_{trip}$, More Stable than $\mathscr{L}_c$

Less sensitive to sampling because always have positive and negative, latent space anchored.



$$\mathscr{L}_{trip}(\,\cdot\,) = \sum_{a,p,n \in B} \underbrace{D(\mathbf{x}^{(a)}, \mathbf{x}^{(p)})}_{\text{make small}} - \underbrace{D(\mathbf{x}^{(a)}, \mathbf{x}^{(n)})}_{\text{make large}} + \underbrace{\alpha}_{\text{margin}}$$

Triplet, 3 points:
a=anchor
p=positive
n=negative

Schroff et al. FaceNet: …, 2015, https://arxiv.org/pdf/1503.03832

$$D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \|f_\theta(\mathbf{x}^{(i)}) - f_\theta(\mathbf{x}^{(j)})\|_2$$

More on Triplet Loss

- Triplet Mining:
  - Want “hard” triplet, or results might not be optimal
  - Finding the hardest examples is also hard, requiring exhaustive search
  - Online triplet (opportunistic): if a hard example comes up, add it to hard list

Table 3. Comparison with the state-of-art on the cars-196 and Stanford products.

	Cars-196						Stanford Online Products			
R@	1	2	4	8	16	32	1	10	100	100
HDC	73.7	83.2	89.5	93.8	96.7	98.4	69.5	84.4	92.8	97.7
BIER	78.0	85.8	91.1	95.1	97.3	98.7	72.7	86.5	94.0	98.0
Baseline	79.2	87.2	92.1	95.2	97.3	98.6	72.6	86.2	93.8	98.0
HTL(depth=16)	**81.4**	**88.0**	**92.7**	**95.7**	**97.4**	**99.0**	**74.8**	**88.3**	**94.8**	**98.4**

W. Ge, W. Huang, D. Dong, M. R. Scott, Deep Metric Learning with Hierarchical Triplet Loss, ECCV 2018

Multi-modal Review

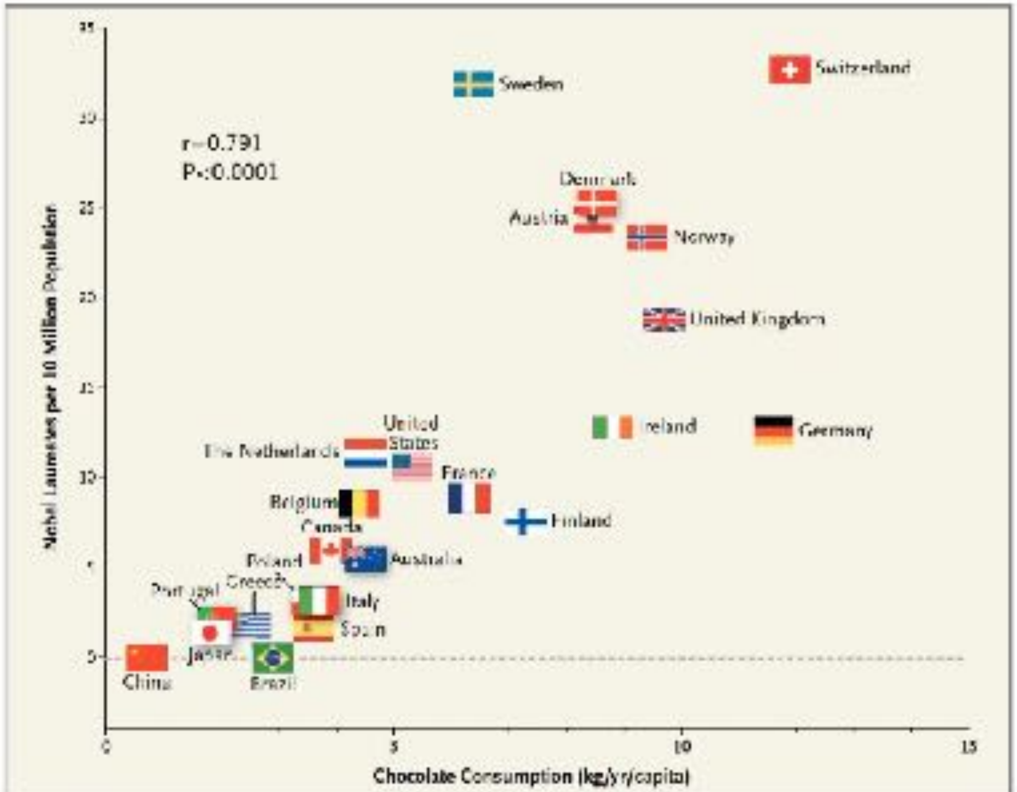


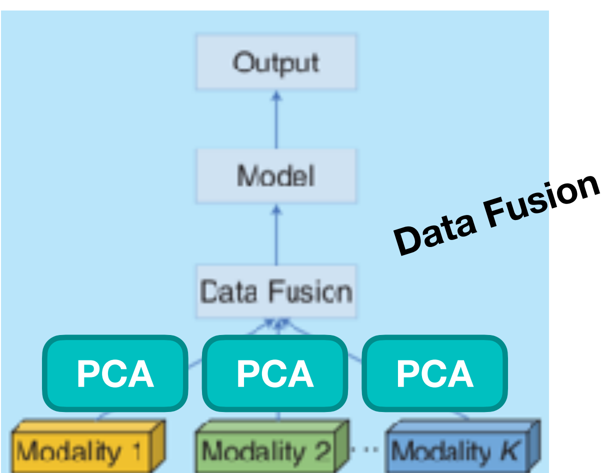
Figure 1. Correlation between Countries' Annual Per Capita Chocolate Consumption and the Number of Nobel Laureates per 10 Million Population.

Multi-modal == Multiple Data Sources

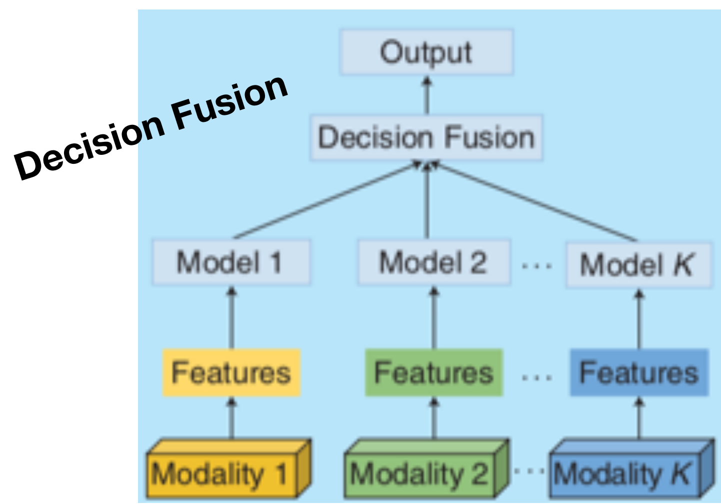
- **Modal** comes from the "sensor fusion" definition from Lahat, Adali, and Jutten (2015) for deep learning
- Using the Keras functional API, this is extremely easy to implement
  - … and we have used it since CS7324!
- But now let's take a deeper dive and ask:
  - What are the different types of modalities that we might try?
  - Is there a more optimal way to merge information?
  - When? Early, Intermediate, and late fusion

Early and Late Stage Fusion

- **Early Fusion:** Merge sensor layers early in the process
- **Assumption**: there is some data redundancy, but modes are conditionally dependent
- **Problem**: architecture parameter explosion
  - Typically need dimensionality reduction



- **Late Fusion**: Merge sensor layers right before flattening
- Use Decision Fusion on outputs
- **Assumption**: little redundancy or conditional independence—just an ensemble architecture
- **Problem**: just separate classifiers, limited interplay

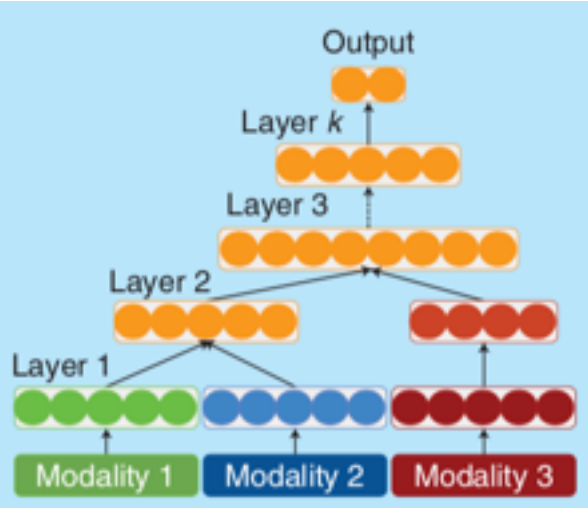


Ramamchandran and Taylor, 2017

Intermediate Fusion

- Merge sensor layers in soft way
- **Assumption**: some features interplay and others do not
- **Problem**: how to optimally tie layers together?

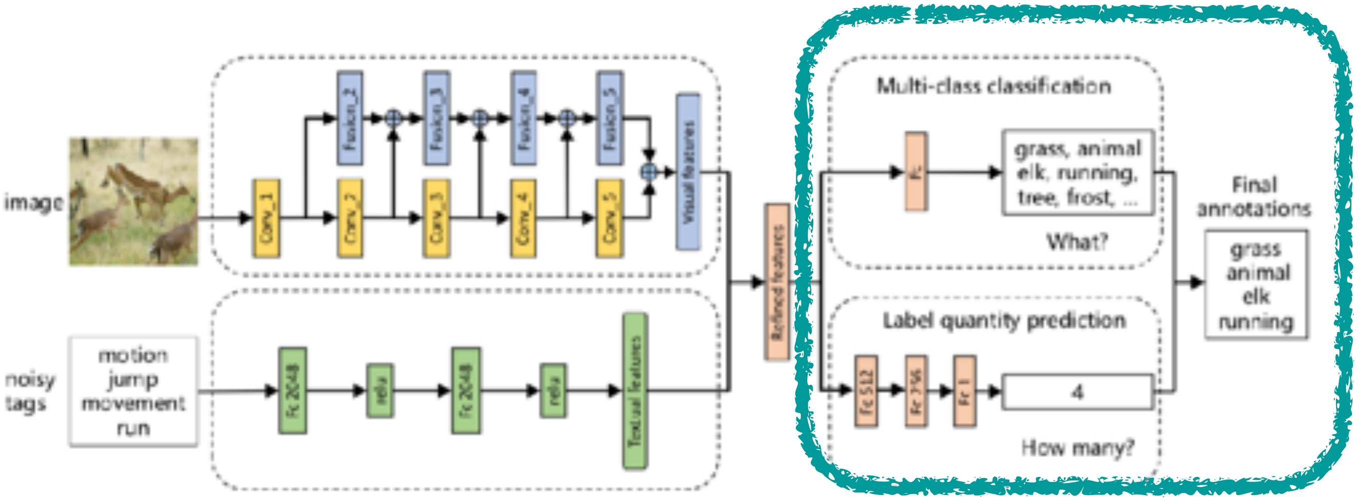
1. Stacked Auto-Encoders [Ding and Tao, 2015]
2. Early fuse layers that are correlated [Neverova *et al.* 2016]
3. Fully train each modality merge based on criterion of similarity in activations [Lu and Xu 2018]
4. Granger Cluster data in each modality and combine [Sylvester *et al.* 2023]



Ramamchandran and Taylor, 2017

Multi-modal Merging

- **Still an open research problem**
- How to develop merging techniques that
  - Can handle exponentially many pairs of modalities
  - Automatically merge meaningful modes
  - Discard poor pairings
  - Selectively merge early or late (or dynamically)

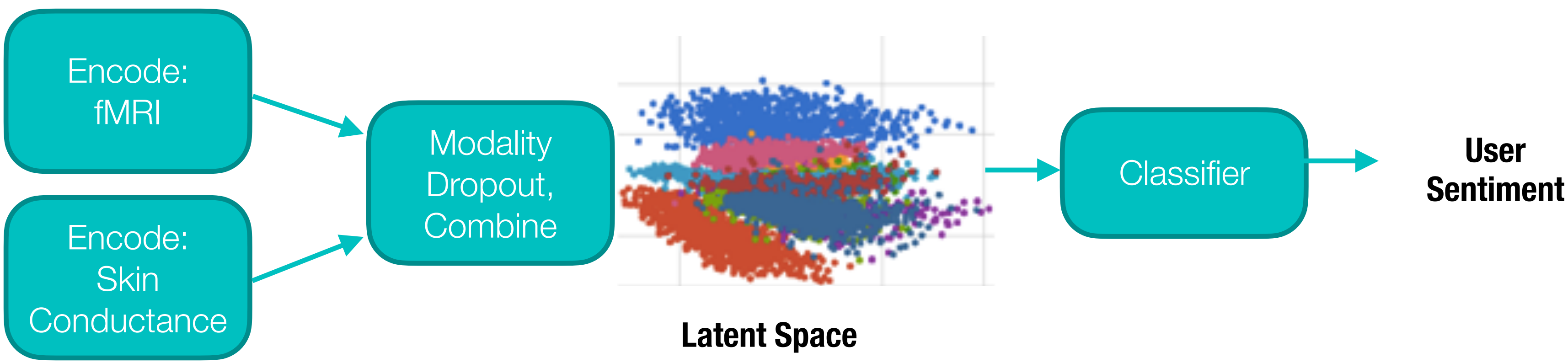


Most current methods are still ad-hoc

https://arxiv.org/pdf/1709.01220.pdf

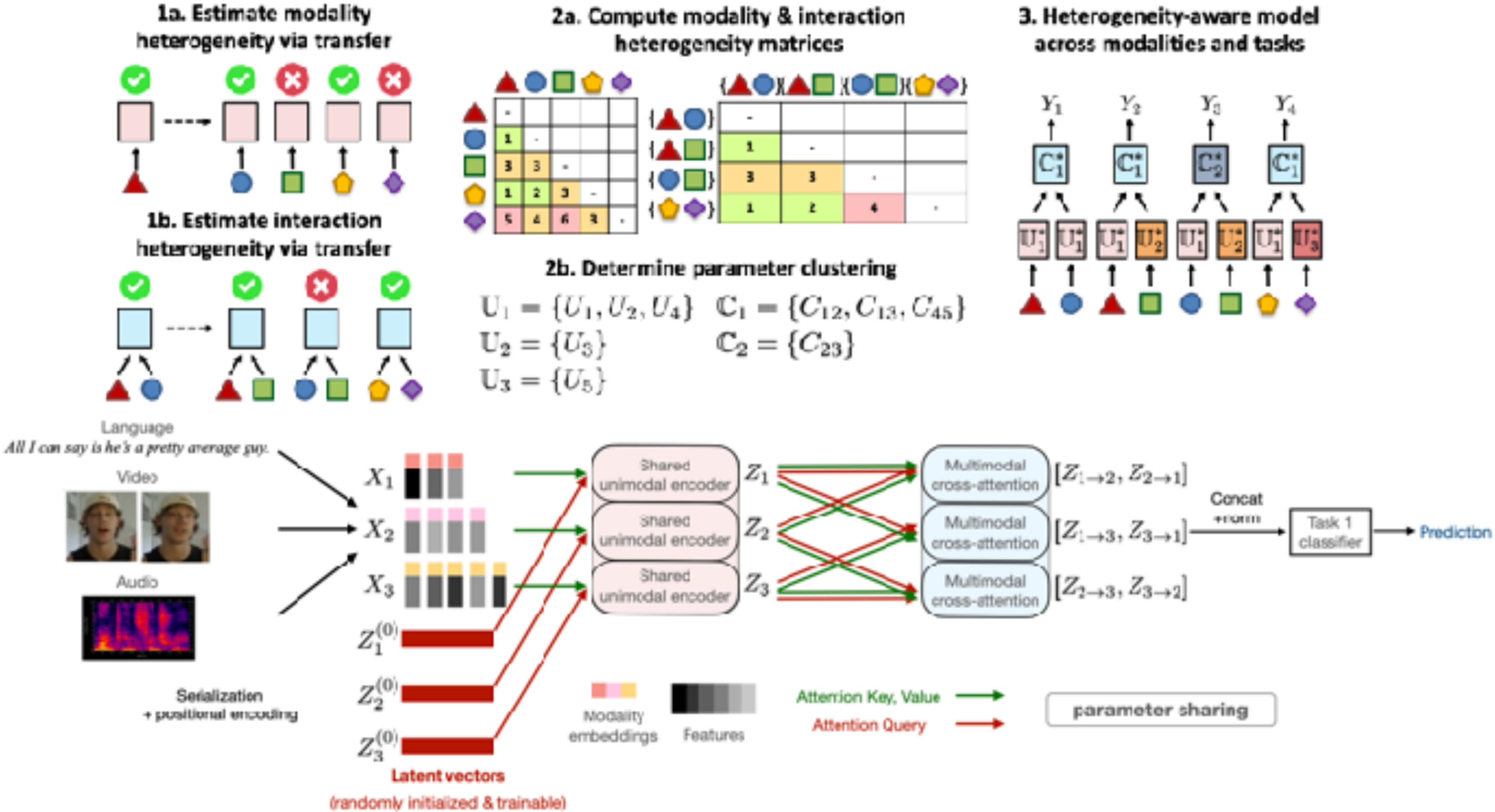
Approaches with Deep Learning

- Latent Space Transfer (universality)
  - From another domain, map to a similar latent space for the same task
  - Useful for unifying data based upon a new input mode when old mode is well understood
    - for example, biometric data
    - **2019-2023, I have never seen a research paper on this…**

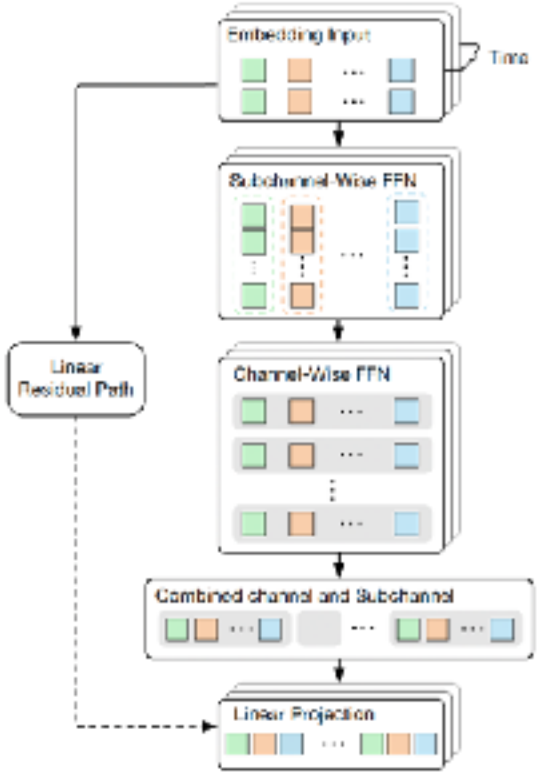


High-Modality Multimodal Transformer: Quantifying Modality & Interaction Heterogeneity for High-Modality Representation Learning

Paul Pu Liang[1], Yiwei Lyu[2], Xiang Fan[1], Jeffrey Tsaw[1], Yudong Liu[1], Shentong Mo[1], Dani Yogatama[3], Louis-Philippe Morency[1], Ruslan Salakhutdinov[1]
[1]Carnegie Mellon University, [2]University of Michigan, [3]DeepMind



Currently in Review



$X \in \mathbb{R}^{T\times C_f\times C_s}$: The input for a single modality
$Y = \mathrm{FFN}_{\text{subchannel}}(X) \in \mathbb{R}^{T\times C_f\times C_{ds}}$: Maps each subchannel to a latent space of higher dimension.
$Z = \mathrm{FFN}_{\text{channel}}(Y) \in \mathbb{R}^{T\times C_{dm}\times C_{ds}}$: Reduces dimensionality of latent representations for each channel.
$E = \mathrm{Linear}(Z) \in \mathbb{R}^{T\times C_{dm}}$: Aggregates the features across the channel and subchannel dimensions to produce the final embedding.

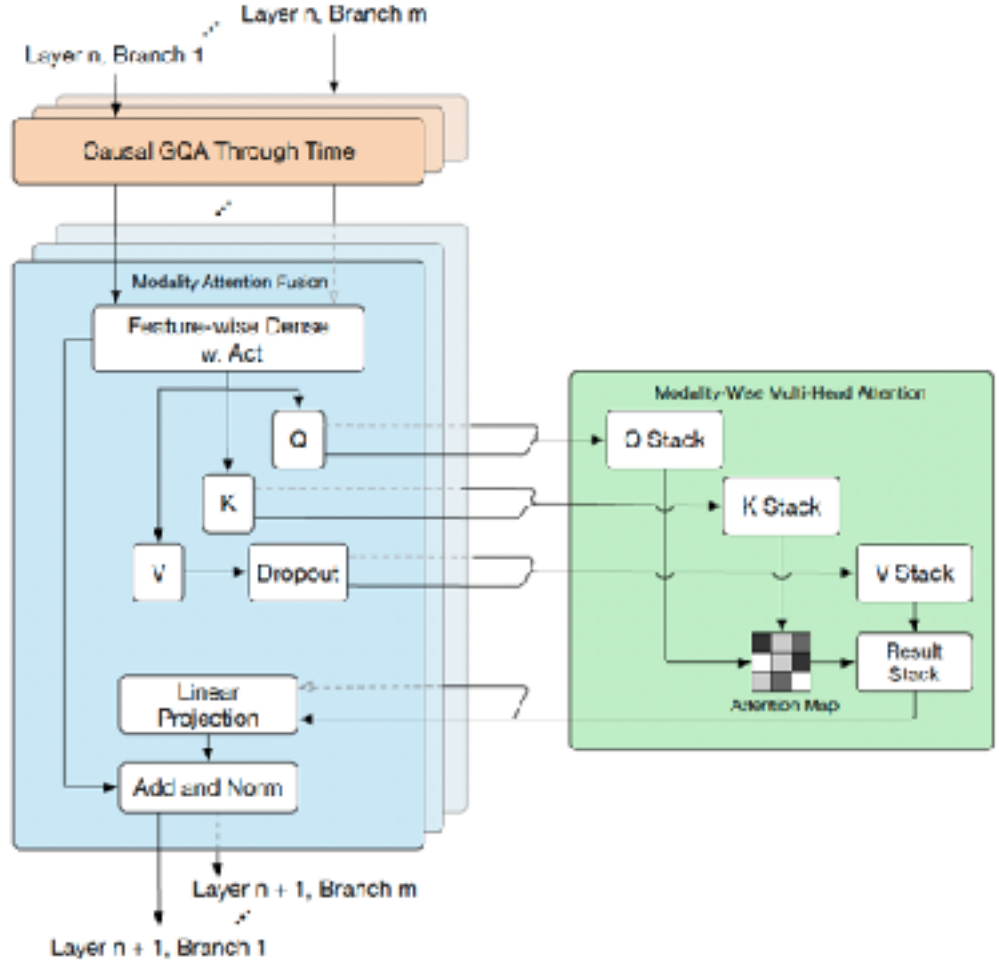
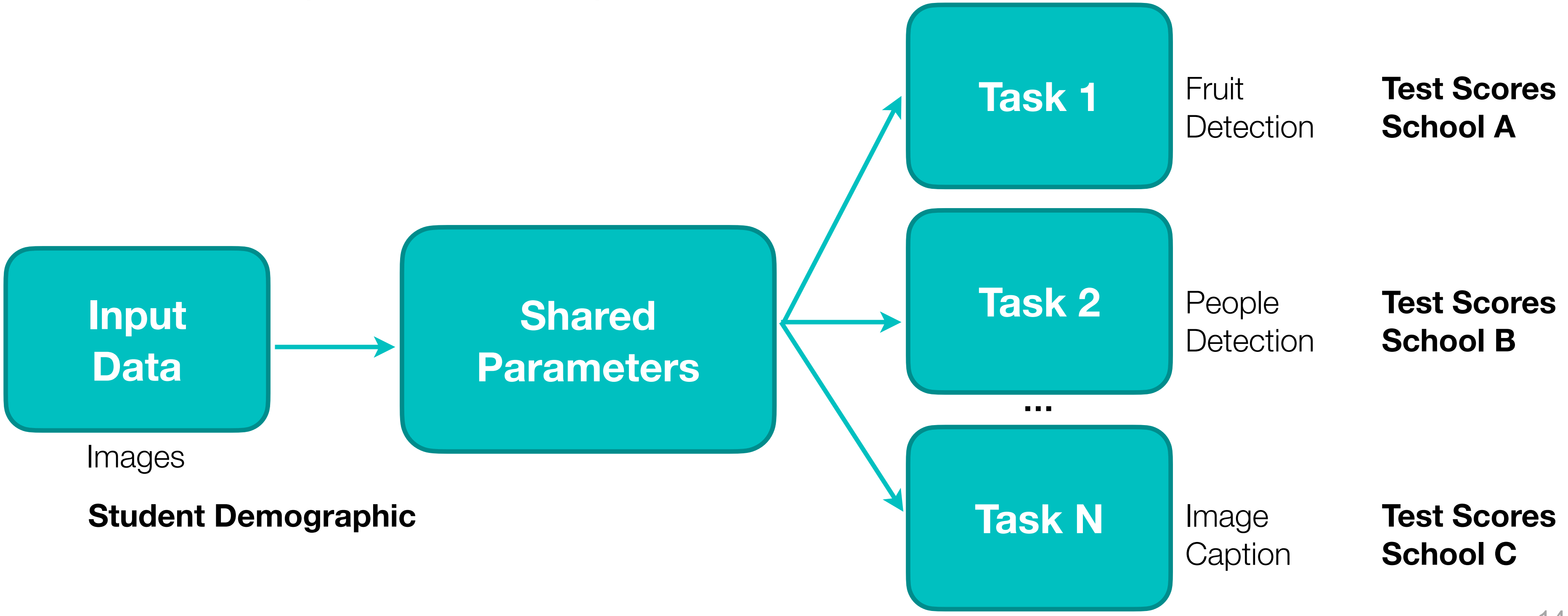


Figure 3. Repetitive Cross-modal Fusion Transformer (RCFT)

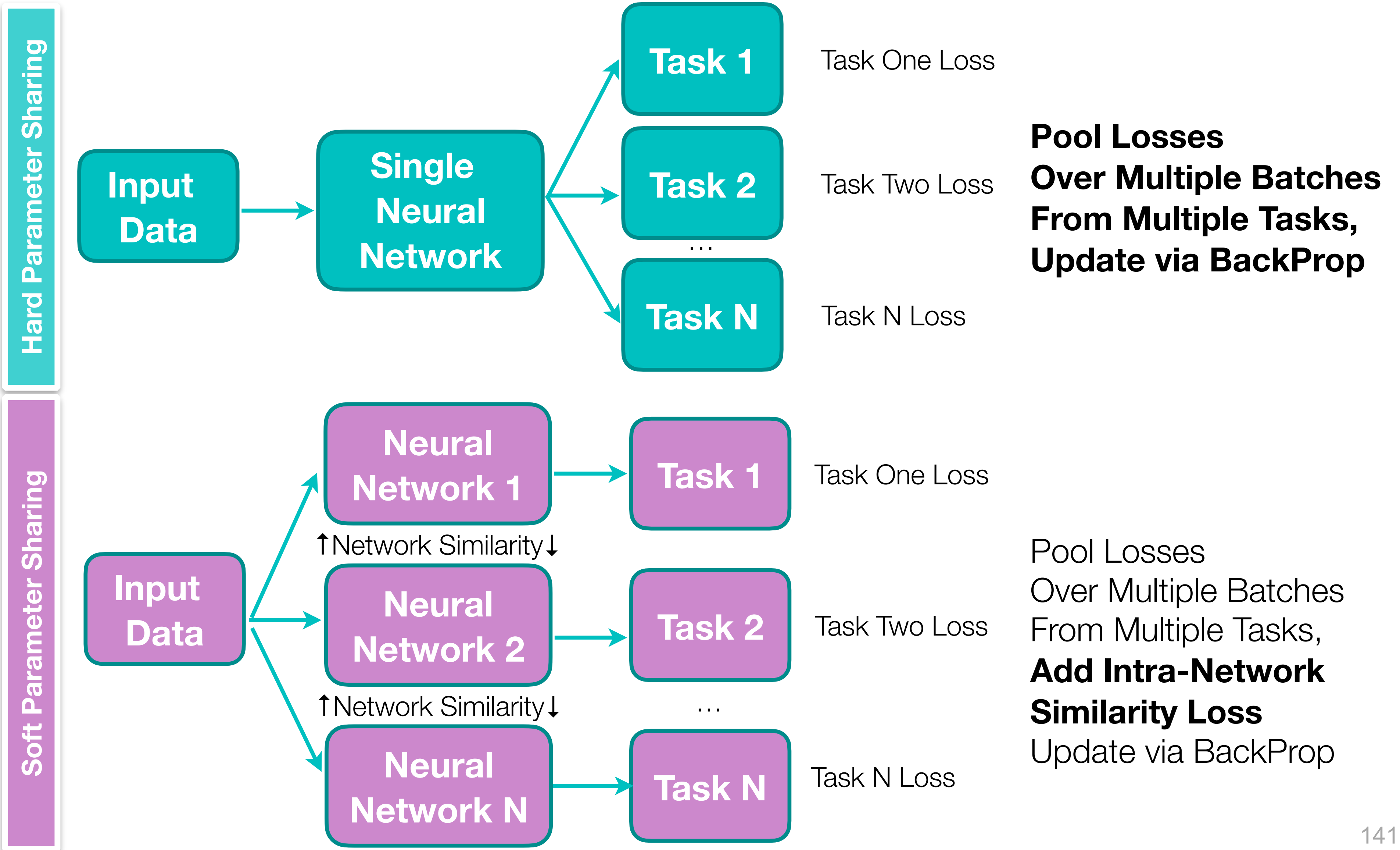
Multi-Task Models

Multi-task learning overview

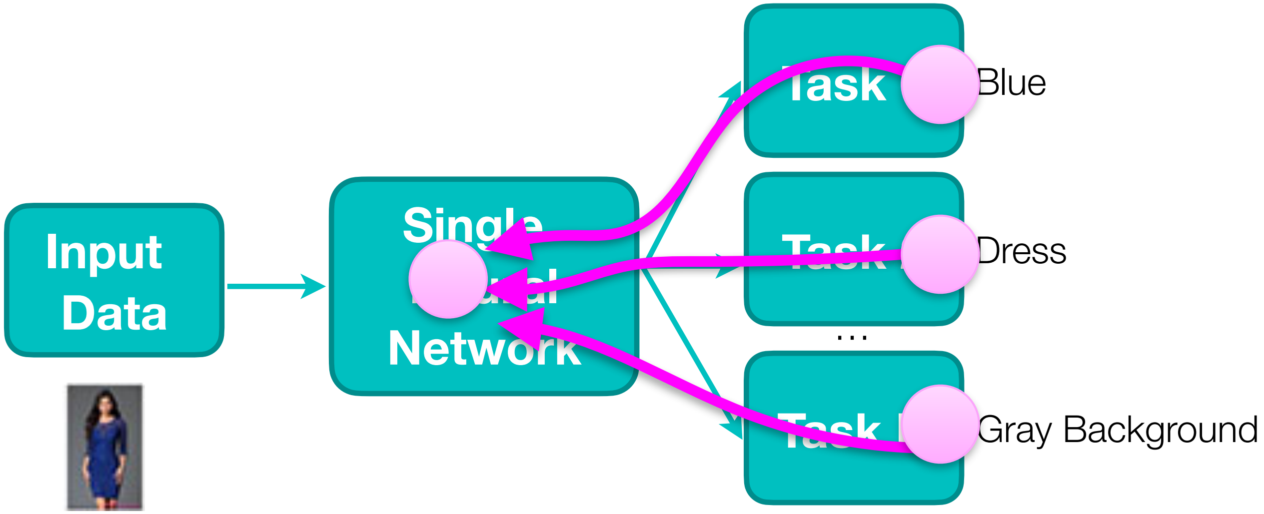
- For deep networks, simple idea: share parameters in early layers
- Used shared parameters as feature extractors
- Train separate, unique layers for each task



Multi-task Learning Parameter Sharing



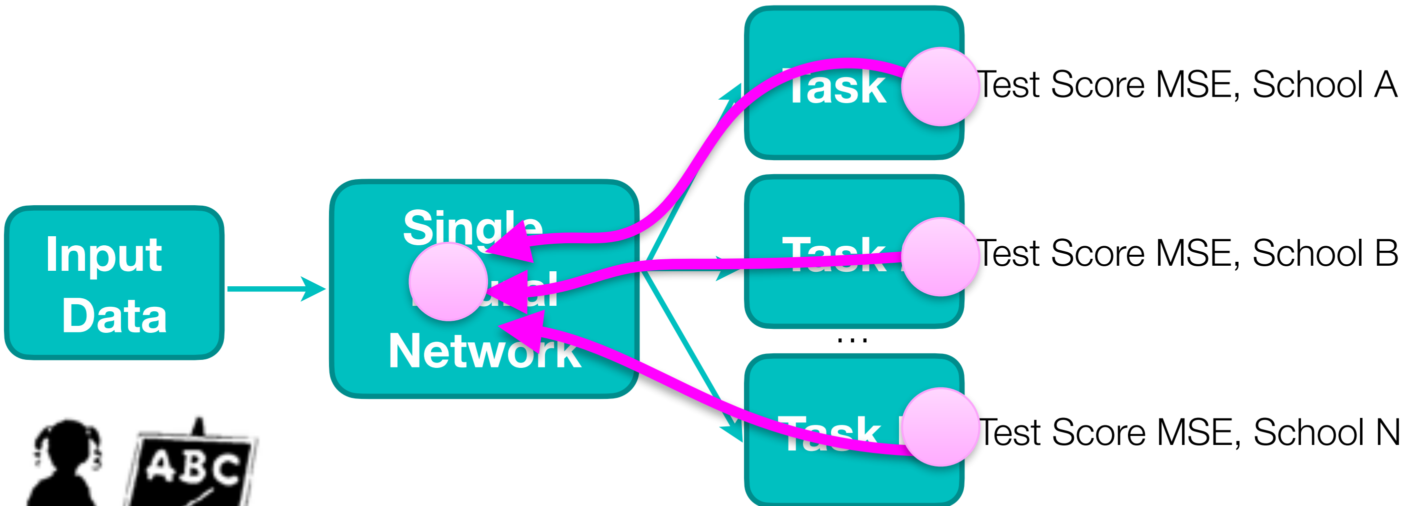
Multi-Label per Input



Measure Loss **for each label simultaneously**
Back propagate **everything at one time** for a given batch

Multi-task Optimization

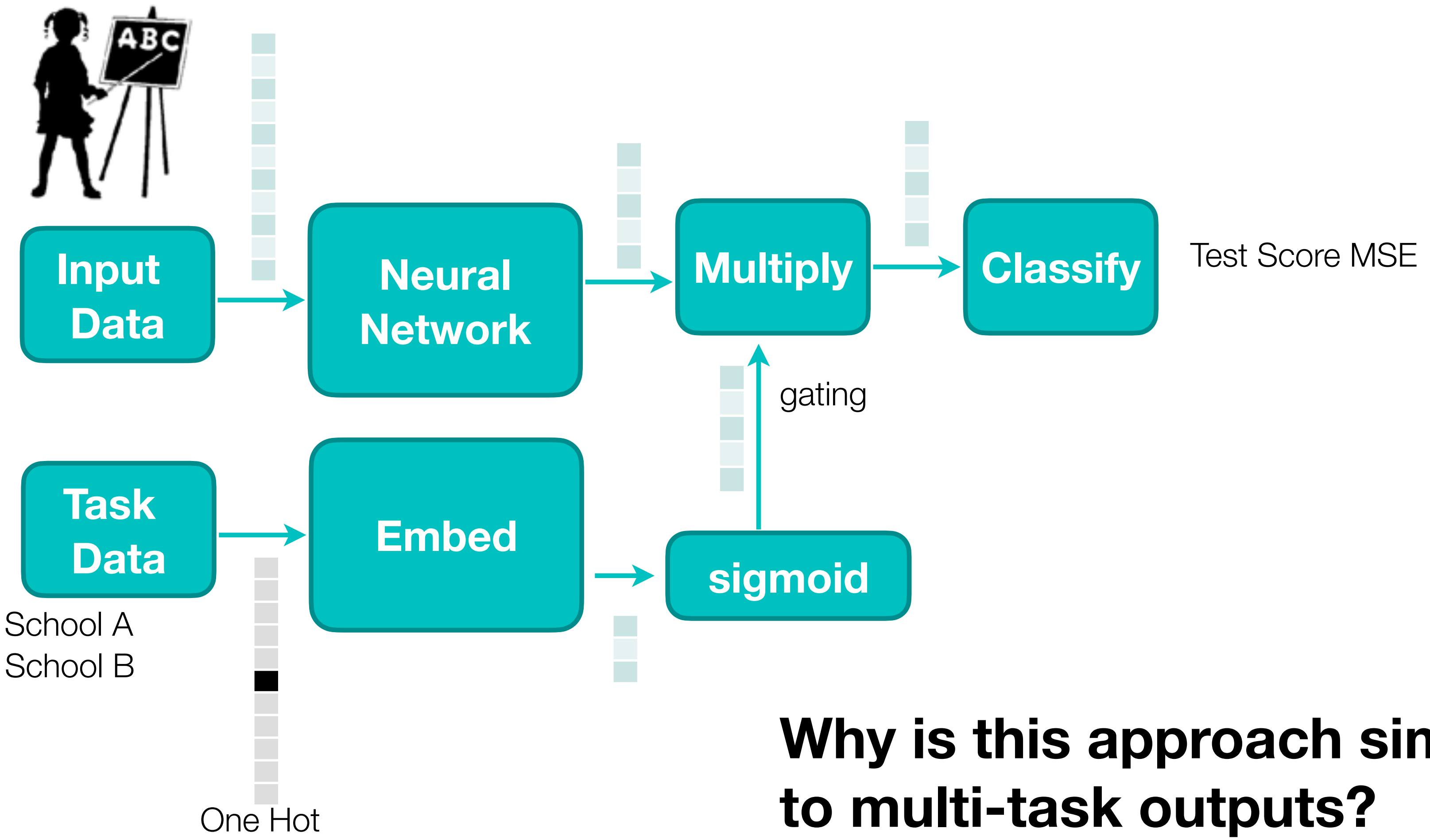
Single Task Label per Input



Method One: Batch updates across multiple tasks
need to perform customized gradient calculations

Method Two: Update small batches using a random task
easier, but can cause instability in training

An alternative: Task-Gating



Why is this approach similar to multi-task outputs?

Multi-Task Learning
School Data, Computer Surveys


Traian-Pop Traian Pop


Sponsor Follow

LukeWood Luke Wood

KerasCV Author, Full Time Keras team member & Machine Learning researcher @ Google, Part Time UCSD Ph.D student

PRO

Method One: Batch updates across multiple tasks
need to perform customized gradient calculations

Method Two: Update small batches using a random task
easier, but can cause instability in training

Follow Along: `LectureNotesMaster/03 LectureMultiTask.ipynb`

Lecture Notes for

Neural Networks and Machine Learning

Loss Multi-Modal and Multi-Task

Next Time:
Circuits

Reading: Chollet 8.1-8.5