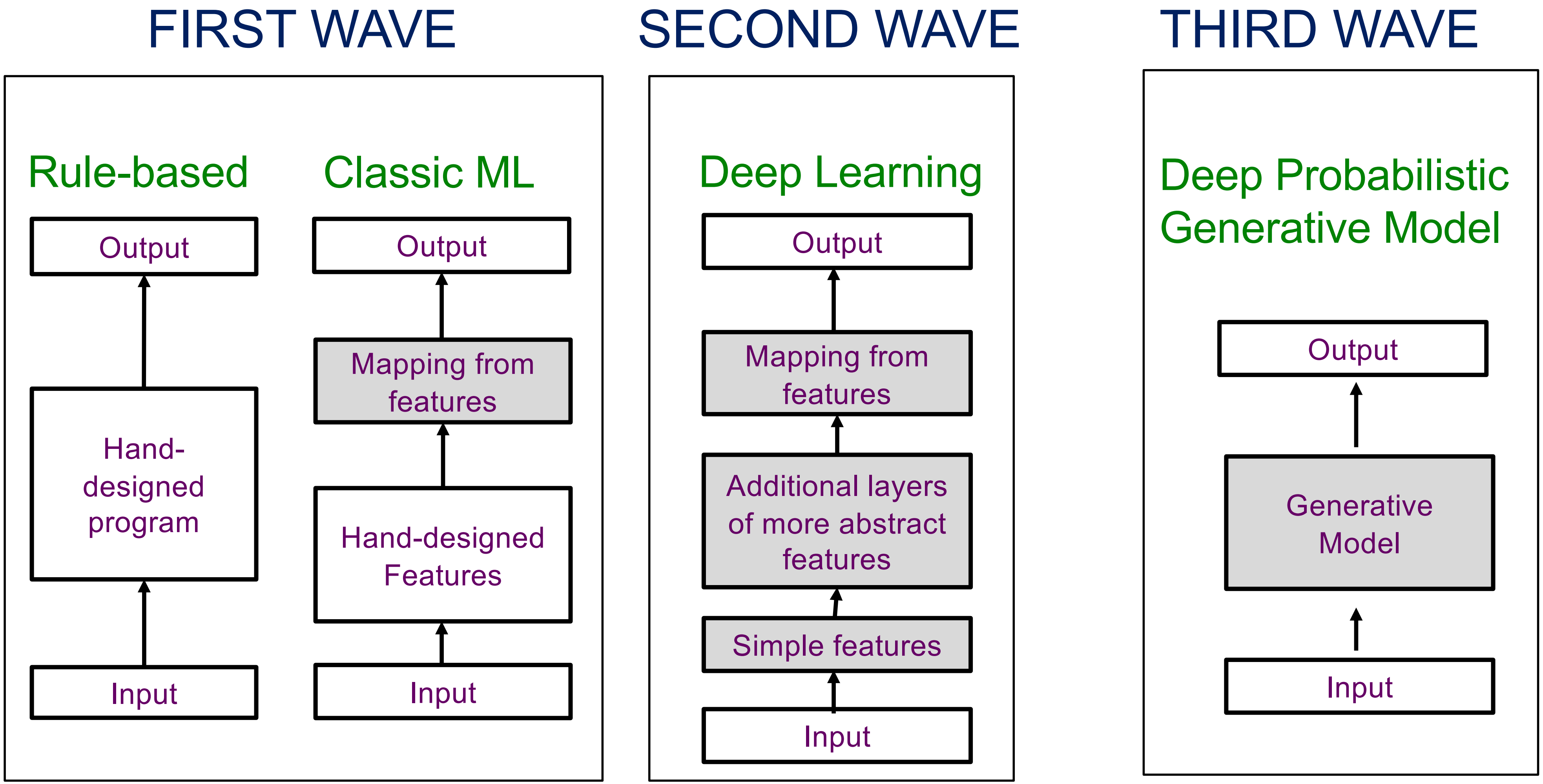
Causal Explanation

Sargur N. Srihari
srihari@buffalo.edu

Topics in Causal Explanation

- Three waves of AI
- Explainable AI
- Probabilistic Graphical Models
- Causal Modeling
- Most Probable Explanation
- Explaning Forensic Evidence

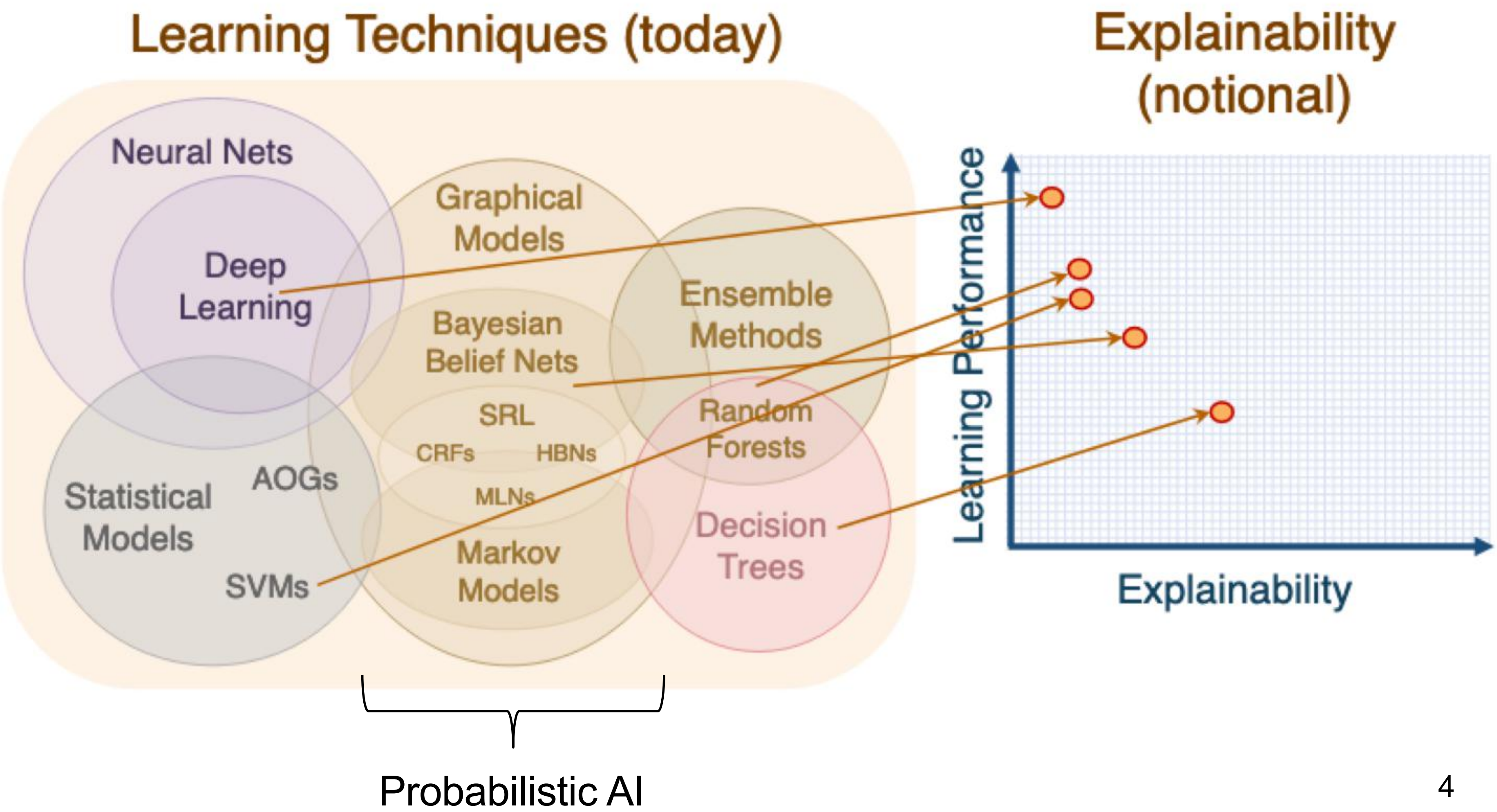
Role of probability in explanation



Naturally suited for XAI

Most Probable Explanation

Explainability of AI Models

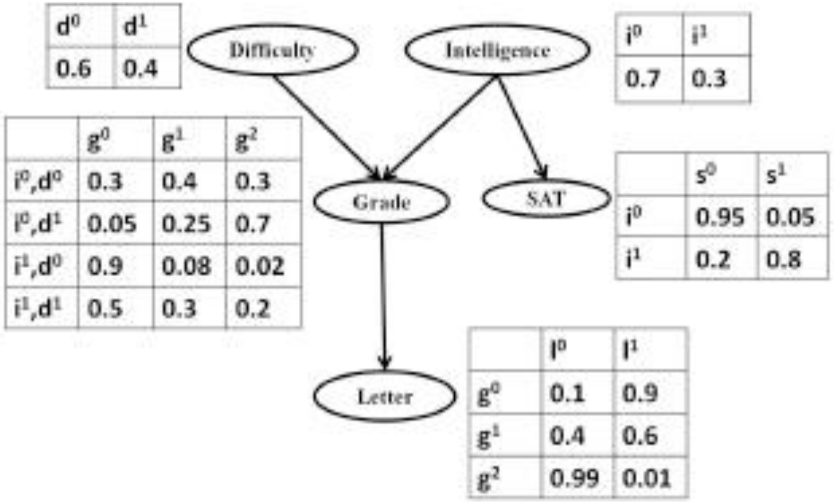


Probabilistic Models in Explanation

- Generative models are useful for
  - Querying
    - BNs, MRFs
  - Sampling
    - VAEs, GANs
- Inference methods useful as explanation
  - Most Probable Explanation (MPE)
  - Causal Inference
  - Explaining Away

Probabilistic Graphical Models (PGMs)

Bayesian Networks



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^0	g^1	g^2
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^1,d^0	0.9	0.08	0.02
i^1,d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

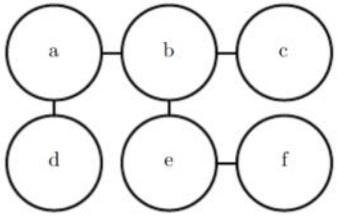
	l^0	l^1
g^0	0.1	0.9
g^1	0.4	0.6
g^2	0.99	0.01

- CPDs: $p(x_i \mid pa(x_i))$
- Joint Distribution $\boldsymbol{x} = \{x_1, .. x_n\}$

$$\boxed{p(\boldsymbol{x}) = \prod_{i=1}^{N} p(x_i \mid pa(x_i))}$$

$$P(D,I,G,S,L) =$$
$$P(D)P(I)P(G \mid D,I)P(S \mid I)P(L \mid G)$$

Markov Networks



$$p(\mathbf{x}) = \frac{1}{Z}\hat{p}(\mathbf{x}) \qquad \tilde{p}(\mathbf{x}) = \prod_{C \in G} \phi(C)$$

$$Z = \int \tilde{p}(\mathbf{x})\, d\mathbf{x}$$

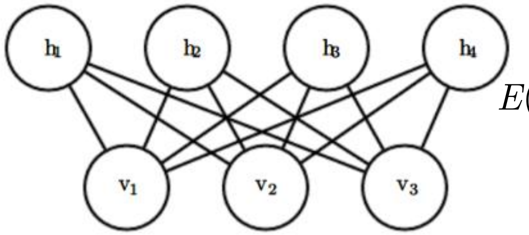
$$\boxed{p(a,b,c,d,e,f) = \frac{1}{Z}\phi_{a,b}(a,b)\phi_{b,c}(b,c)\phi_{a,d}(a,d)\phi_{b,e}(b,e)\phi_{e,f}(e,f)}$$

Energy model $E(a,b,c,d,e,f) =$

$E_{a,b}(a,b) + E_{b,c}(b,c) + E_{a,d}(a,d) + E_{b,e}(b,e) + E_{e,f}(e,f)$

$\phi_{a,b}(a,b) = \exp(-E(a,b))$

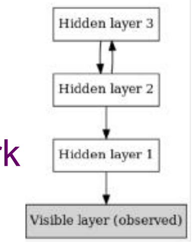
Restricted Boltzmann machine



$$E(\boldsymbol{v},\boldsymbol{h}) = -\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{h} - \boldsymbol{v}^{\mathrm{T}}\mathrm{W}\boldsymbol{h}$$

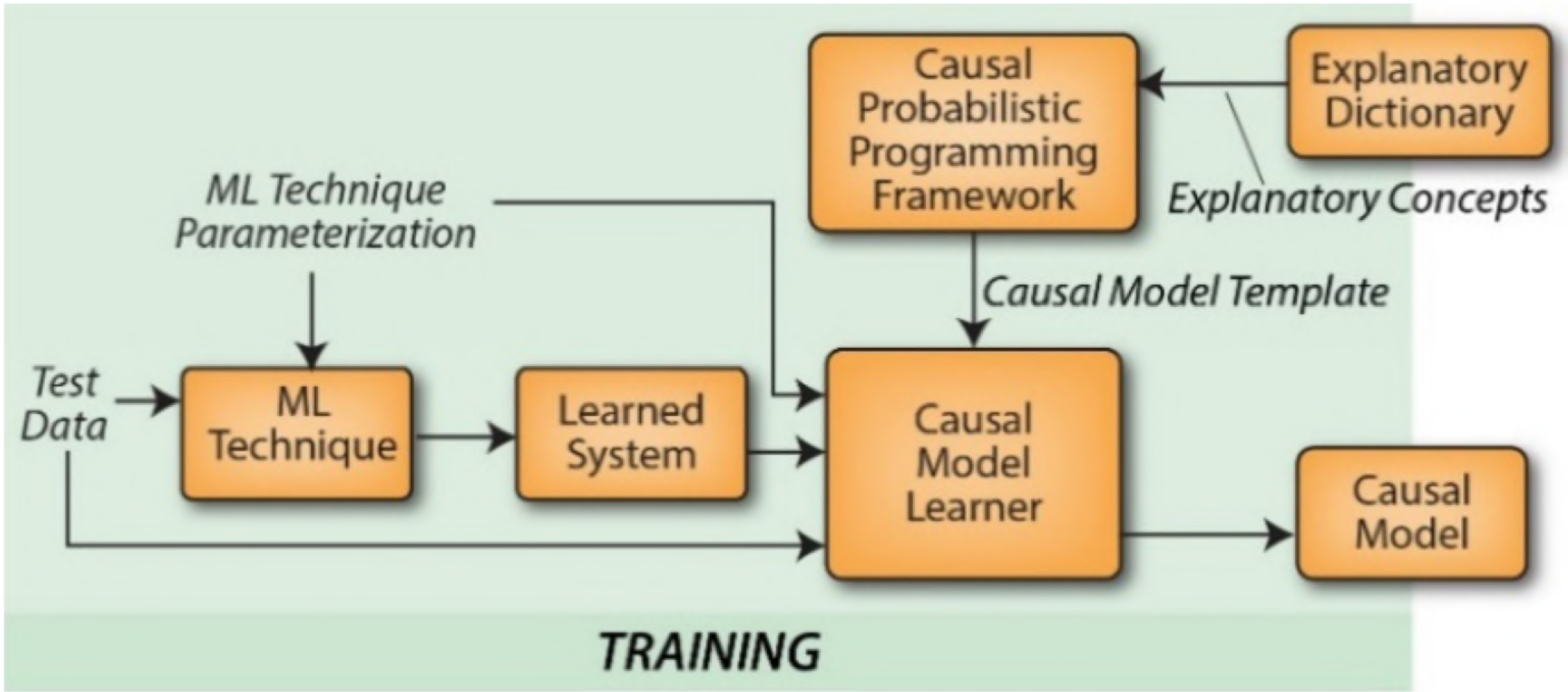
$p(\boldsymbol{h}|\boldsymbol{v}) = \Pi_i p(h_i|\boldsymbol{v})$ and
$p(\boldsymbol{v}|\boldsymbol{h}) = \Pi_i p(v_i|\boldsymbol{h})$

Deep Belief Network



XAI from Causal Model Induction

- Experiment with the learned model to learn an explainable, causal, probabilistic programming model



Causal Modeling

- For a G defining a causal directed PGM on variables $X = x_1, \ldots, x_\mathrm{n}$, the joint distribution is

$$P(x_1, ..., x_n) = \prod_i P(x_i | pa_i)$$

  - where pa_i are the parents of x_i
  - An intervention on G by setting $x_i = x'_i$, denoted as $do(x'_i)$, induces a modified graph G' where the edges from pa_i to x_i are removed, resulting in post intervention distribution

$$P(x_1, ..., x_n | do(x'_i)) = \begin{cases} \prod_{j \neq i} P(x_j | pa_j) & \text{if } x_i = x'_i \\ 0 & \text{if } x_i \neq x'_i \end{cases}$$

- Semantics of causality are important for explaining DNNs, because explanations must be causal models

Explanation of DNN decision

- When one seeks an explanation of a network's decisions, one is equivalently asking ”What changes can be made to the input for the output to change or stay the same?”
- Consider a causal model that defines the joint distribution $P(\mathbb{O},\mathbb{P},\mathbb{X})$ over a set of DNN outputs $\mathbb{O}$, inputs $\mathbb{P}$, & intermediate variables $\mathbb{X}$
- Explanation of an observed DNN output is formulated as intervention

Causal Representation in DNNs

- A user could ask counterfactual questions about the network, i.e. $P(\mathbb{O},\mathbb{P},\mathbb{X}\ |\mathrm{do}(x'_i))$ for any input, output, or internal neuron in the network
- But it serves poorly as a model of explanation
  - due to the lack of human–level concepts that underlie any arbitrary neuron in the network:
    - saying neuron i caused the network to detect a pedestrian may be correct but does not satisfy human needing explanations
- Thus, a DNN causal model must be at a granularity meaningful to humans.

Mapping Neurons to Concepts

- A causal model for DNNs should be represented by joint distribution over $\mathbb{O}$,$\mathbb{P}$, and a set of concepts $\mathbb{C}$
- The process for deriving $\mathbb{C}$ is described by a function $f_{\mathbb{R}}$: $\mathbb{R} \longrightarrow \mathbb{C}$ over a specific DNN that transforms the representation of neurons and their activations into a set of concept variables
- Ideally, $f_{\mathbb{R}}$ would have the following properties:

$$\int_R P(\mathbb{O}, \mathbb{P}, \mathbb{R}) = \int_C P(\mathbb{O}, \mathbb{P}, \mathbb{C})$$

$$P(\mathbb{O}, \mathbb{P} | R, do(p_i')) = P(\mathbb{O}, \mathbb{P} | C, do(p_i'))$$

Concept Extraction

- One way to construct concepts that satisfy the semantics is to consider network activations
- But concepts needn't be restricted to those
  1. Concepts should be few to minimize amount of investigation a human would need to employ
  2. Concepts should be interpretable
     - in the case of images, we would like activations to be restricted to contiguous areas containing consistent, interpretable visual features
  3. Concepts should contain all relevent information needed for achieving network task

Auxiliary neural network

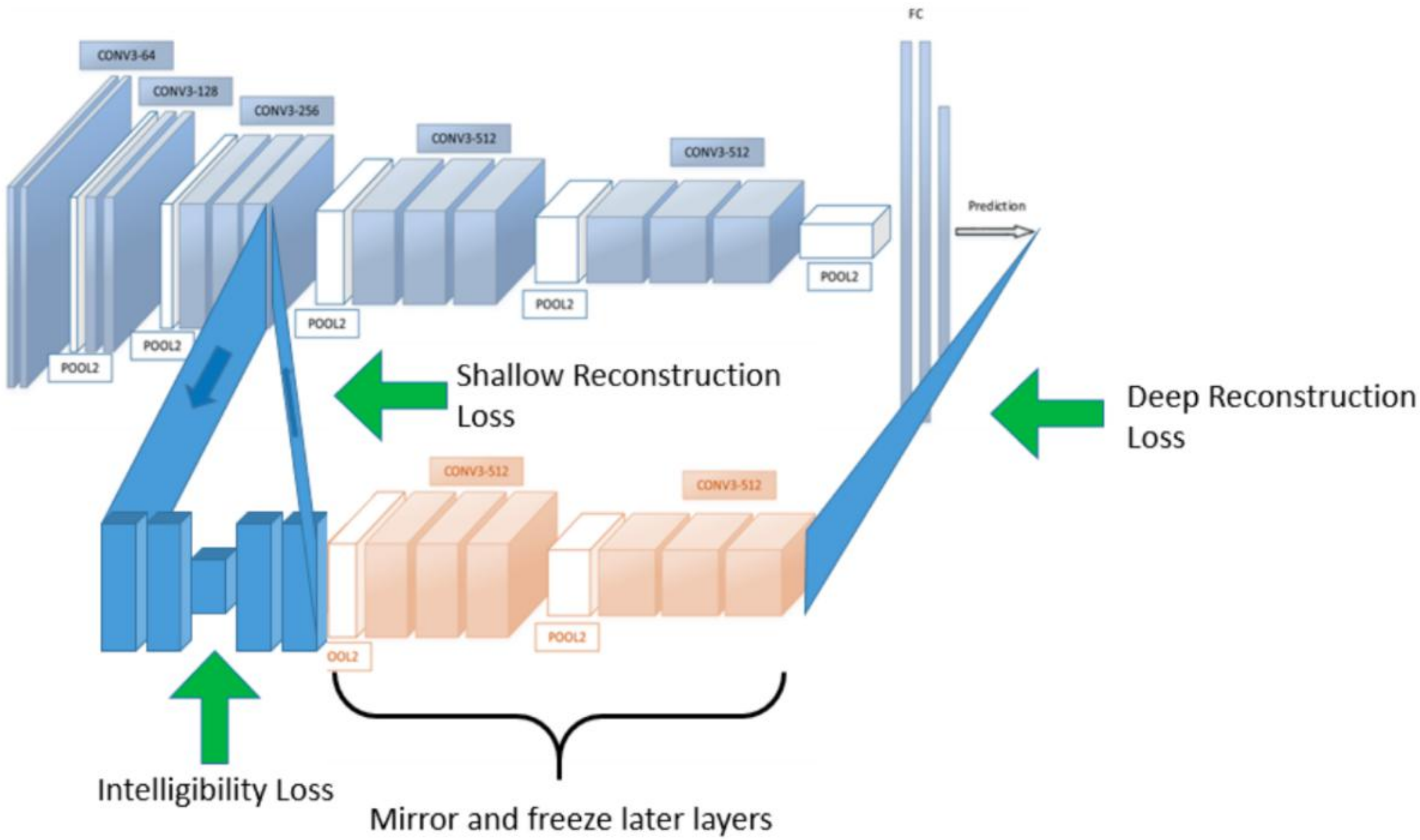
- Network constructs concept representations satisfying above properties
- Using an autoencoder with two loss functions
  - Shallow reconstruction loss: L_1 norm between input and output activations

$$L_{shallow}(\theta; a_i) = |d_\theta(c_\theta(a_i)) - a_i|_1$$
$$L_{deep}(\theta; a_i) = KL(r(a_i)||r(d_\theta(c_\theta(a_i))))$$

  - Deep reconstruction loss: reconstructed activations result in same classification output after being passed through the rest of network
  - Total autoencoder loss

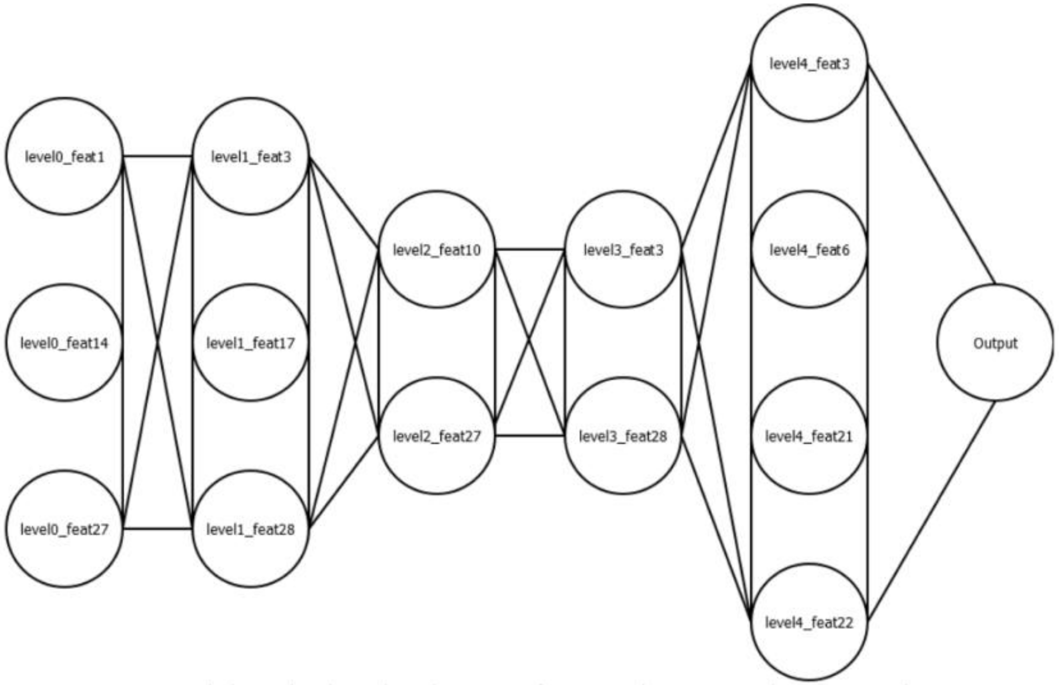
$$\begin{aligned} L(\theta; x_i) = &\lambda_{shallow} L_{shallow}(\theta; x_i) + \\ &\lambda_{deep} L_{deep}(\theta; x_i) + \\ &\lambda_{interpretability} L_{interpretability}(\theta; x_i) \end{aligned}$$

Autoencoder for a single layer



Learned causal Bayes Net

- For Inria pedestrian data set



Crossed boxes indicate that all nodes of a given level have edges incident on each of the nodes of the subsequent level.

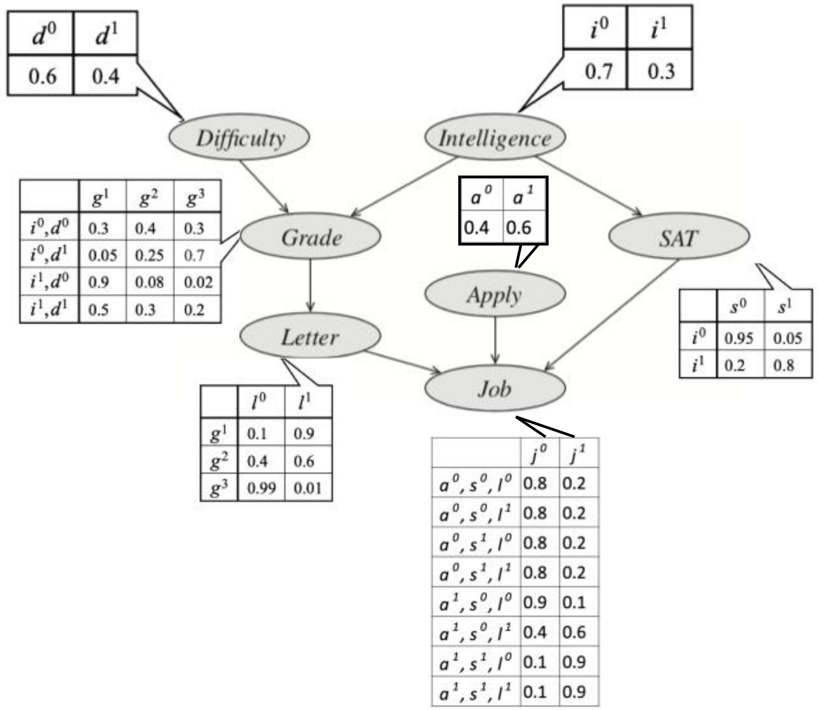
Summary of Causal Learning XAI

- Approach to explain predictions to relate output prediction to concepts represented within
  - Series of autoencoders encouraging interpretable properties to construct concepts in activations throughout network
  - Pool features and intervene on autoencoded network to construct variables used to build a causal BN
  - Use network to identify features of causal relevance to individual classifications which are then visualized

Most Probable Explanation (MPE)

Observe evidence: student has received a job offer $J = j^1$.
We want an explanation for each of the following two queries

1. Most probable (MAP) setting for all variables I, D, G, S, L, A
2. Most probable setting for a particular variable I



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0, d^0	0.3	0.4	0.3
i^0, d^1	0.05	0.25	0.7
i^1, d^0	0.9	0.08	0.02
i^1, d^1	0.5	0.3	0.2

a^0	a^1
0.4	0.6

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

	j^0	j^1
a^0, s^0, l^0	0.8	0.2
a^0, s^0, l^1	0.8	0.2
a^0, s^1, l^0	0.8	0.2
a^0, s^1, l^1	0.8	0.2
a^1, s^0, l^0	0.9	0.1
a^1, s^0, l^1	0.4	0.6
a^1, s^1, l^0	0.1	0.9
a^1, s^1, l^1	0.1	0.9

$$P(I, D, G, S, L, A, J) = P(I)P(D)P(G|I, D)P(S|I)P(L|G)P(A)P(J|A, S, L)$$

Query 1: Inferring all variables

Conditioning with Job with $J = j^1$

Conditional Probability:

$$P(I, D, G, S, L, A \mid J = j^1) = \frac{P(I,D,G,S,L,A,J = j^1)}{P(J = j^1)}$$

Numerator:

$$P(I, D, G, S, L, A, J = j^1) \quad = P(I)P(D)P(G|D, I)P(L|G)P(S|I)P(A)P(J = j^1|\mathrm{A, S, L})$$

Denominator requires Marginal Inference

$$P(J = j^1) = \sum_I \sum_D \sum_G \sum_S \sum_L \sum_A P(I, D, G, S, L, A, J = j^1) = 0.3679$$

MPE for all variables

$$P(I,D,G,S,L,A \mid J = j^1) = \frac{P(I,D,G,S,L,A,J=j^1)}{P(J=j^1)}$$

Exhaustive listing of 96 Combinations of numerator with probabilities

Sr. No.	Difficult	Intelligen	Grade	Letter	SAT	Apply	Job	Joint Probabil
3	0	1	0	1	1	1	1	0.0629856
5	0	0	0	1	0	1	1	0.0387828
10	0	0	1	1	0	1	1	0.0344736
13	1	1	0	1	1	1	1	0.023328
16	1	0	2	0	0	0	1	0.01474704
18	1	0	1	1	0	1	1	0.014364
21	1	0	2	0	0	1	1	0.01106028
22	0	1	0	1	0	1	1	0.0104976
23	1	1	2	0	1	1	1	0.01026432
25	0	0	2	0	0	0	1	0.00948024
27	0	1	0	1	1	0	1	0.0093312
28	1	1	1	1	1	1	1	0.0093312
29	0	0	0	1	0	0	1	0.0086184
31	0	0	1	1	0	0	1	0.0076608
32	0	0	2	0	0	1	1	0.00711018
33	0	1	0	0	1	1	1	0.0069984
37	1	1	1	0	1	1	1	0.0062208

Max of numerator (a joint probability):

$$P(I,D,G,S,L,A, J{=}j^1) = 0.9 * 0.6 * 0.8 * 0.9 * 0.9 * 0.3 * 0.6 = 0.0629856$$

Max Conditional Probability (MAP):

$$P(I, D, G, S, L, A|J = 1) = 0.0629856/0.3679 = 0.1712$$

MPE is:

- $I = i^1$: Is *Intelligent*
- $D = d^0$: course is *Difficult*
- $G = g^0$: *Grade is A*
- $S = s^1$: high *SAT*
- $L = l^1$: strong *Letter*
- $A = a^1$: *Apply* for Job

Query 2: Marginal for I given $J = j^1$

The most probable explanation of $Intelligence$ given Job : $J = j^1$

Answer: $I = i^1$: Is $Intelligent$

MAP of Student Intelligence

```
+---------+-------+
| intel_0 | 0.381 |
+---------+-------+
| intel_1 | 0.619 |
+---------+-------+
```

Formula:

$$P(I|J=1) = P(I,J)/P(J=1)$$

$$P(I,J) = P(I)P(J|A,S,L)P(A)P(S|I)P(L|G)P(G|D,I)P(D)$$

$$P(I=0, J=1) = 0.9 * 0.6 * 0.8 * 0.9 * 0.9 * 0.3 * 0.6 = 0.0629$$

$$P(I=1, J=1) = 0.6 * 0.7 * 0.3 * 0.9 * 0.95 * 0.6 * 0.6 = 0.0387$$

$$P(I=0|J=1) = 0.0387/(0.0629 + 0.0387) = 0.381$$

$$P(I=1|J=1) = 0.0629/(0.0629 + 0.0387) = 0.619$$

Compare to Prior of Student Intelligence

i^0	i^1
0.7	0.3

MPE is NP-Hard

- BN is an MRF whose factors are conditional probabilities and $Z=1$
- MAP inference is simpler than Marginal Inference since we can ignore Z

Marginal $\boxed{p(y=1) = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_n} p(y=1, x_1, x_2, \ldots, x_n)}$ ← Needs Partition function

MAP $\boxed{\max_{x_1,\ldots,x_n} p(y=1, x_1, \ldots, x_n)}$ ← Can ignore Partition function

- However MAP inference is still not easy
  - Both Marginal and MAP inference are NP-hard

Sampling from Student BN

Algorithm 12.1 Forward Sampling in a Bayesian network

Procedure Forward-Sample (
    $\mathcal{B}$ // Bayesian network over $\mathcal{X}$
)
    Let $X_1, \ldots, X_n$ be a topological ordering of $\mathcal{X}$
    for $i = 1, \ldots, n$
        $\boldsymbol{u}_i \leftarrow \boldsymbol{x}\langle \mathrm{Pa}_{X_i} \rangle$ // Assignment to Pa_{X_i} in $x_1, \ldots, x_{i-1}$
        Sample x_i from $P(X_i \mid \boldsymbol{u}_i)$
    return $(x_1, \ldots, x_n)$

PGMPY:

```python
from pgmpy.sampling import BayesianModelSampling

forward_sampler = BayesianModelSampling(student)
samples = forward_sampler.forward_sample(size=1000)
```

	apply	intel	sat	diff	grade	letter	job
0	1	1	1	1	0	1	1
1	0	0	0	0	0	1	0
2	1	1	1	1	1	1	1
3	1	0	0	1	2	0	0
4	1	1	1	0	2	0	1
5	0	0	0	1	2	0	1
6	1	0	0	0	0	1	0
7	1	0	1	0	1	1	1
8	1	0	0	0	1	1	1
9	1	0	0	1	1	0	0
...					...		
993	0	0	1	1	1	1	0
994	1	1	1	0	0	1	1
995	1	0	0	0	2	0	0
996	0	0	0	0	0	1	0
997	1	0	0	1	2	0	0
998	0	0	0	0	0	0	1
999	1	0	0	0	2	0	0

1000 rows × 7 columns

PGM classification over 1000 samples

```python
from pgmpy.models import BayesianModel
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination
```

```python
student = BayesianModel()
student.add_nodes_from(['diff', 'intel', 'grade','sat','letter','apply','job'])
student.add_edges_from([('diff', 'grade'),
                        ('intel', 'grade'),
                        ('intel', 'sat'),
                        ('grade', 'letter'),
                        ('apply', 'job'),
                        ('letter','job'),
                        ('sat','job')])
student.fit(train)
```

```python
inference = VariableElimination(student)
counter = 0
for index, row in test.iterrows():
    mle1 = inference.map_query(variables={'sat','letter'},
                               evidence={'job': row['job']})
    mle2 = inference.map_query(variables={'intel'},
                               evidence={'sat': mle1['sat'],
                                         'letter': mle1['letter']})
    if(mle2['intel'] == row['intel']):
        counter += 1

print('Accuracy is: ' + str(counter/test.shape[0]*100))
```

```
Accuracy is: 88.125
```

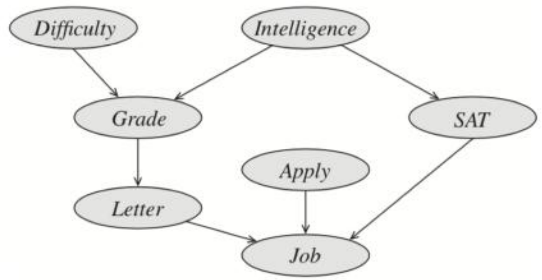


Figure 5.3 The Student example augmented with a *Job* variable

Results:

- Training Accuracy: 92%
- **Testing Accuracy: 88%**

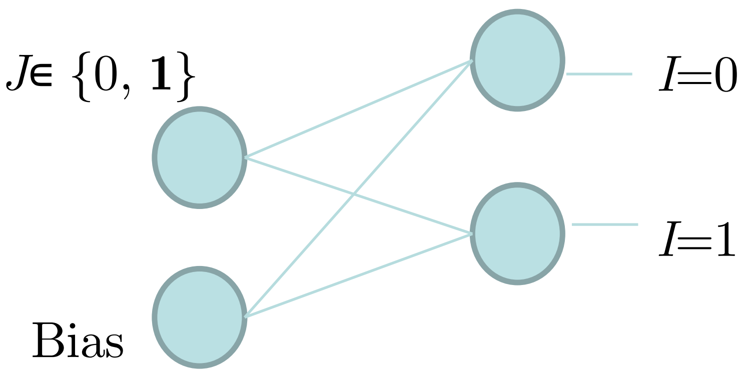
Logistic Regression for Intelligence

Training Hyperparameters:
- Number of Epochs: 100
- Batch Size: 100
- Number of samples: 1000
- Number of Training samples: 800
- Number of Validation samples: 200

Results:
- Training Accuracy: 94.62%
- Training Loss: 0.1580
- **Testing Accuracy: 96%**
- **Testing Loss: 0.1534**

Logistic Regression output:

	Intelligence = 0	Intelligence = 1
Job = 1	0.038	0.962



Explanation:

MPE

evidence: student **gets job**

explanation: 96% probability that student is **intelligent**;

Formula:

$$P(I=0|J=1) = 0.498$$

$$P(I=1|J=1) = 0.502$$

Explainability vs Performance

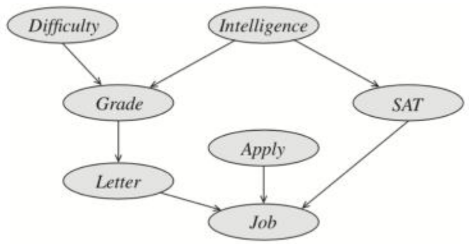


Figure 5.3 The Student example augmented with a *Job* variable

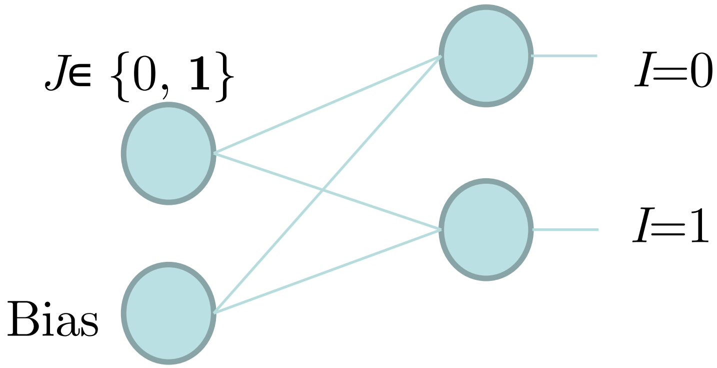
```
inference = VariableElimination(student)
counter = 0
for index, row in test.iterrows():
    mle1 = inference.map_query(variables={'sat','letter'},
                               evidence={'job': row['job']})
    mle2 = inference.map_query(variables={'intel'},
                               evidence={'sat': mle1['sat'],
                                         'letter': mle1['letter']})
    if(mle2['intel'] == row['intel']):
        counter += 1

print('Accuracy is: ' + str(counter/test.shape[0]*100))
```

```
Accuracy is: 88.125
```

Explainable System Results:

- Training Accuracy: 92%
- **Testing Accuracy: 88%**



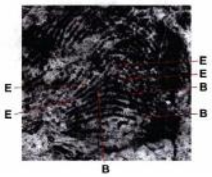
High Performance System Results:

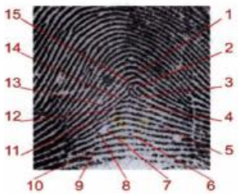
- Training Accuracy: 95%
- **Testing Accuracy: 96%**

Forensic Comparison

- Impression Evidence:
  materials with characteristics of impressed objects
  - Footwear impressions

  - latent fingerprints





  - Handwriting samples

Expert Features

Feature Vector

imagename	0968c_num1.png
pen_pressure	2
letter_spacing	2
size	2
dimension	1
is_lowercase	2
is_continuous	2
slantness	3
tilt	2
entry_stroke_a	1
staff_of_a	2
formation_n	2
staff_of_d	3
exit_stroke_d	2
word_formation	2
constancy	1

Number of Features: 15

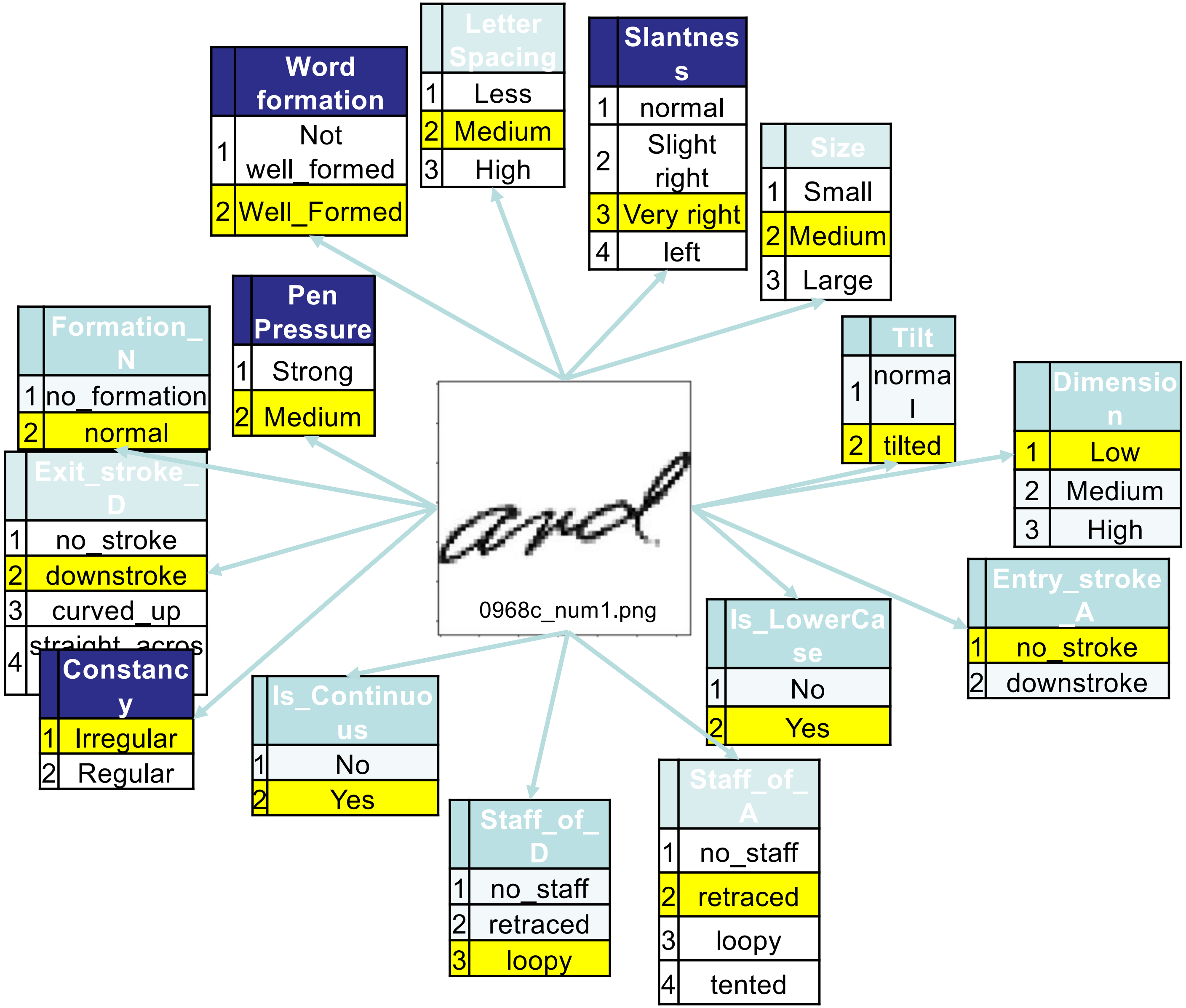
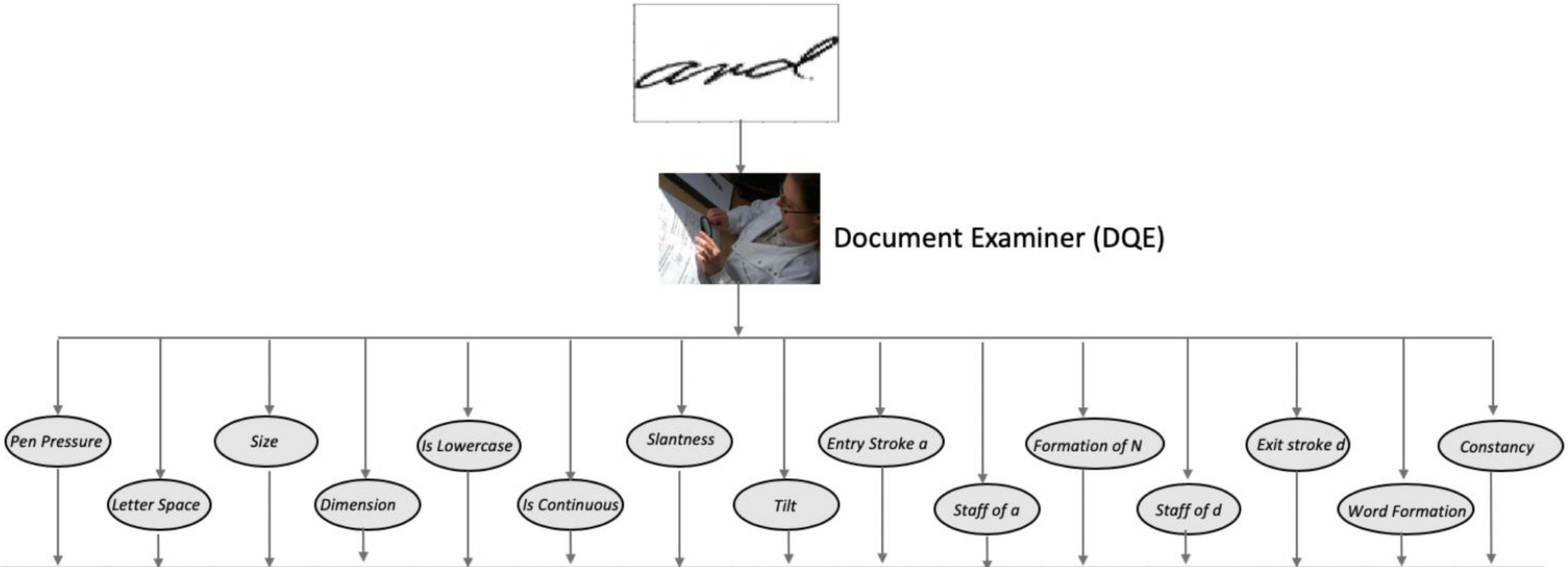


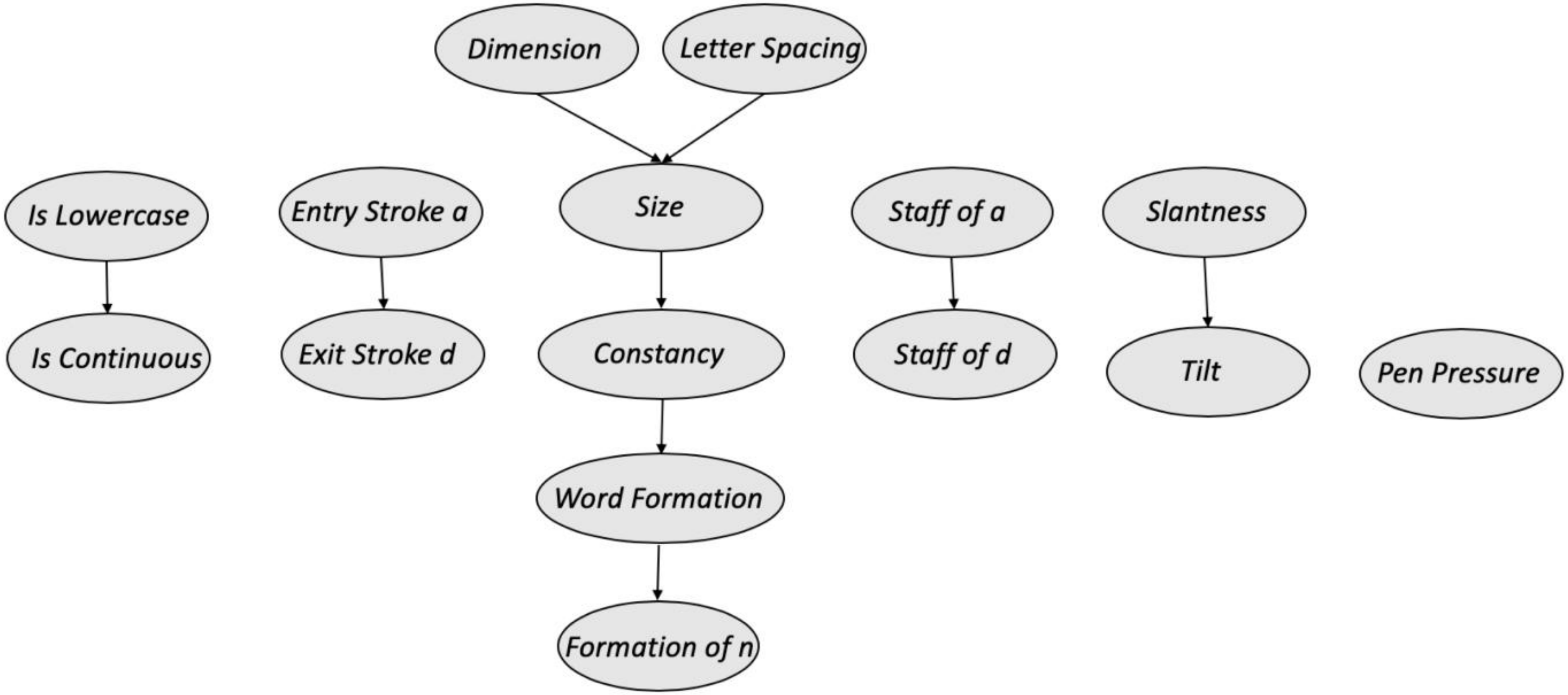
Image 1: Human observed features mapped to image

Handwriting comparison– expert features

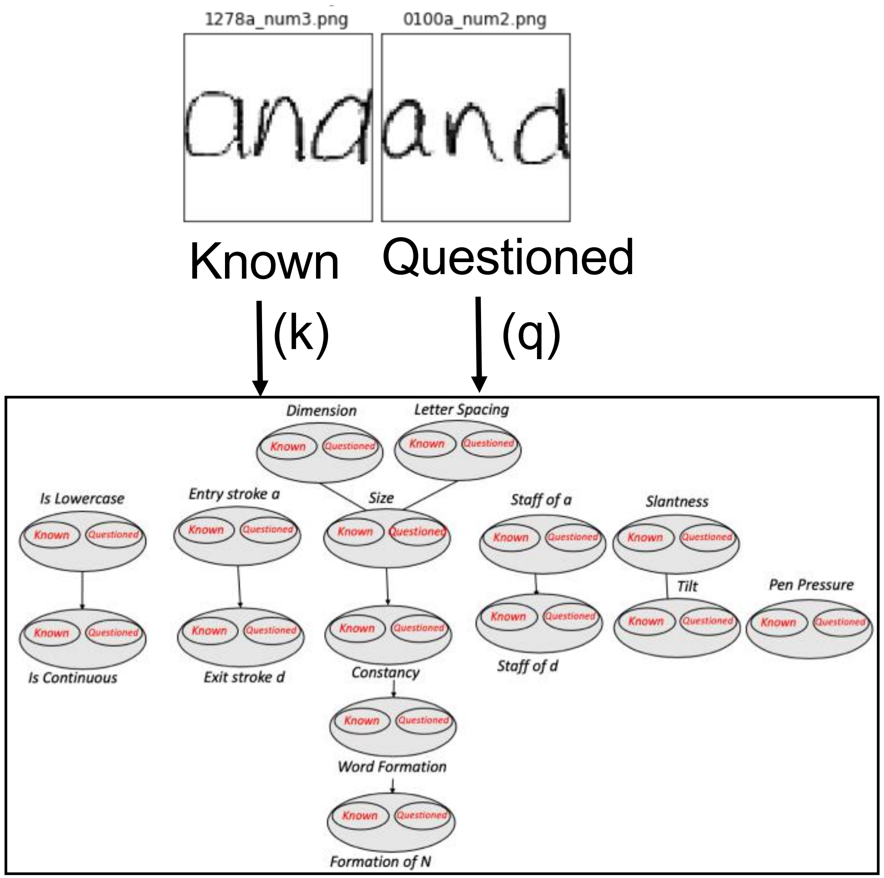


	Pen Pressure	Letter Spacing	Size	Dimension	Is LowerCase	Is Continuous	Slantness	Tilt	Entry stroke "A"	Staff of "A"	Formation "N"	Staff of "D"	Exit stroke "D"	Word format	Constancy
1	Strong	Less	Small	**Low**	No	No	normal	normal	**no stroke**	no staff	no formation	no staff	no stroke	Not well forr	**Irregular**
2	**Medium**	**Medium**	**Medium**	Medium	**Yes**	**Yes**	slight right	**tilted**	downstroke	**retraced**	**normal**	retraced	**downstroke**	**Well Formec**	Regular
3		High	Large	High			**very right**			loopy		**loopy**	curved up		
4							left			tented			straight across		

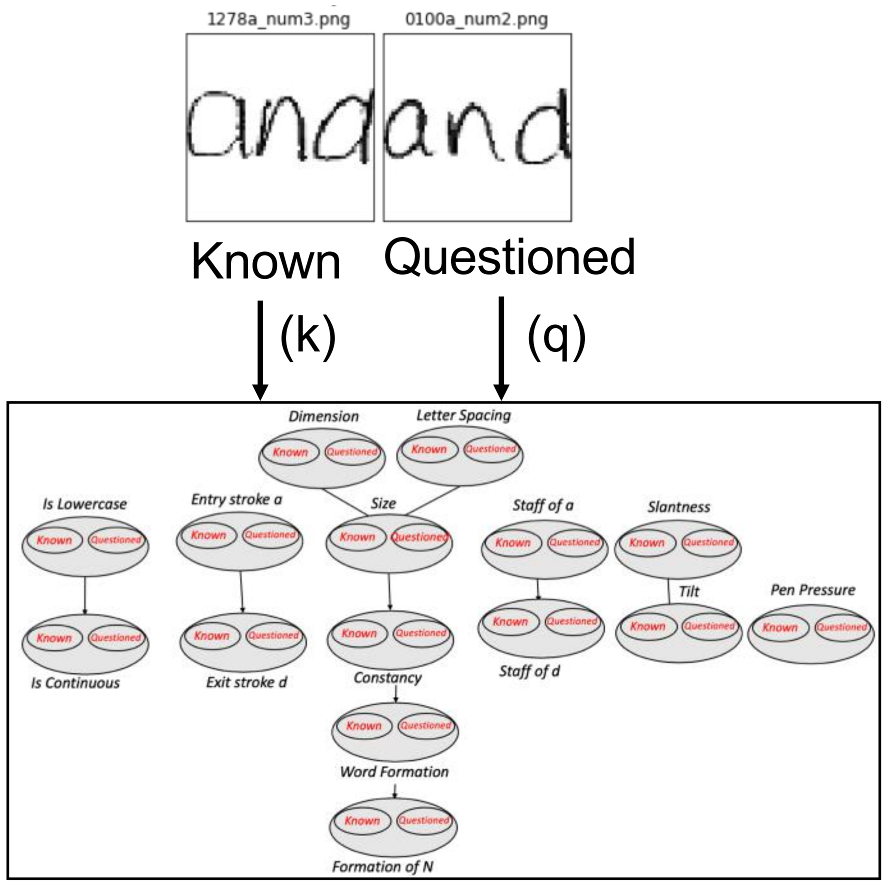
PGM for human observed ‘and’ dataset



PGM for Writer Verification



PGM for same writers (h = 0)



PGM for different writers (h = 1)

$$P(k, q \mid h = 0)$$

$$P(k, q \mid h = 1)$$

$$LLR = \ln\left(\frac{P(k, q \mid h = 0)}{P(k, q \mid h = 1)}\right)$$

Finding LLR

$$LLR = \ln\left(\frac{P(k,q \mid h=0)}{P(k,q \mid h=1)}\right)$$

Approach 1

- A simple **Distance based LLR** approach defines scalar distance between known and questioned samples defining loglikelihood ratio as follows:

$$LLR_D = \ln\left(\frac{P(d(k,q) \mid h=0)}{P(d(k,q) \mid h=1)}\right)$$

Disadvantage: Distribution of N variables is mapped into a scalar thereby incurring severe loss of information (many pairs of k and q can have the same distance).

Approach 2

- **Rarity with Distance based LLR** approach uses strength of the features between the known and the questioned samples defining loglikelihood ratio as follows:

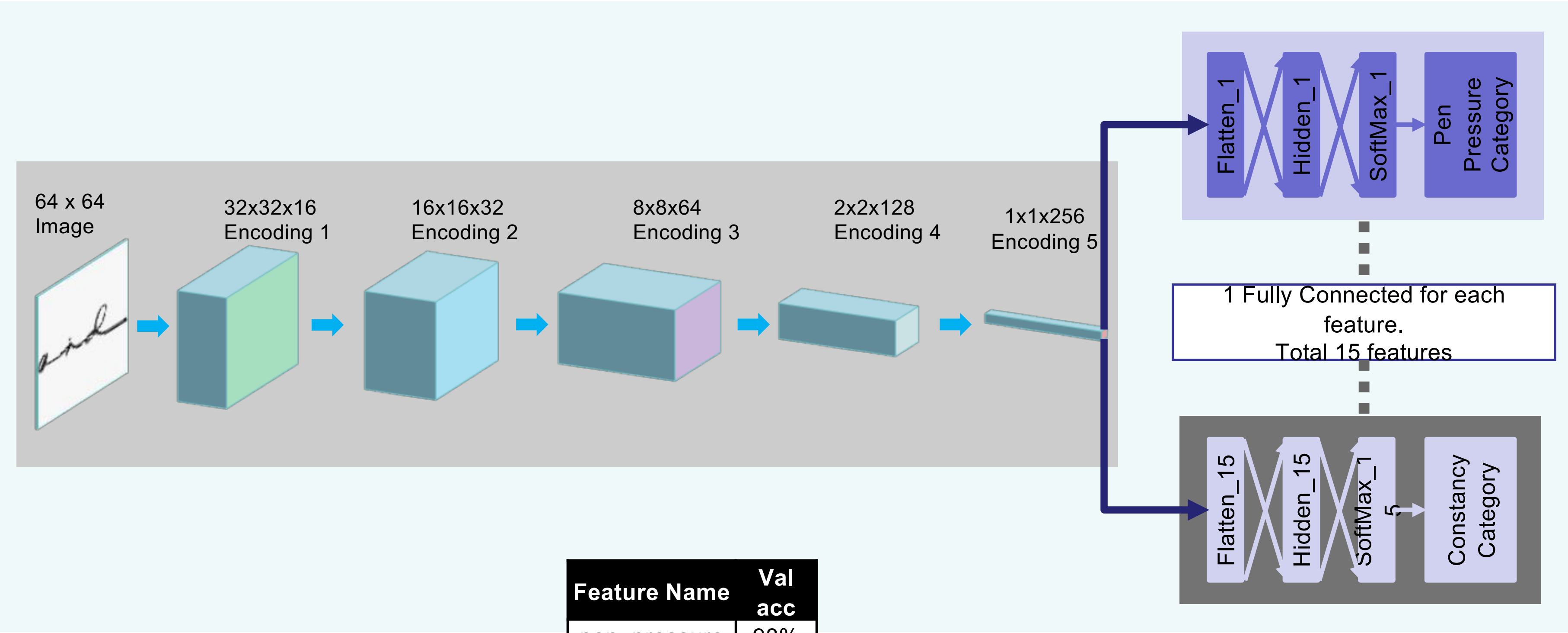
$$LLR_{RD} = \ln\left(\frac{\boxed{\frac{1}{\sqrt{2}\sigma}e^{-\frac{(k-q)^2}{4\sigma^2}}}}{\boxed{\frac{1}{\tau}e^{-\frac{(m-\mu)^2}{4\tau^2}}}}\right)$$

Distance

Rarity

Disadvantage: Not always computationally tractable

Feature extraction architecture/performance



Feature Name	Val acc
pen_pressure	98%
letter_spacing	78%
size	92%
dimension	89%
is_lowercase	99%
is_continuous	94%
slantness	72%
tilt	98%
entry_stroke_a	97%
staff_of_a	84%
formation_n	95%
staff_of_d	86%
exit_stroke_d	66%
word_formation	89%
constancy	84%

Code: https://github.com/mshaikh2/HDL_Forensics

Comparison performance

Performance based on Overall Similarity

Metric	Seen	Unseen	Shuffled
Intra Writer Accuracy	88%	71%	81%
Inter Writer Accuracy	96%	93%	95%
Average Accuracy	96%	94%	95%

Explainability:

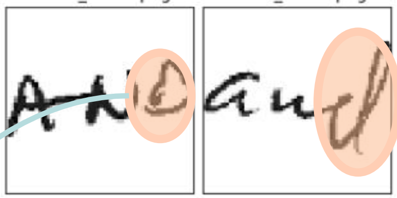
- Compare corresponding 'SoftMax_x' (x : {1 to 15}) of two images using cosine similarity to obtain feature wise explanation

Code: https://github.com/mshaikh2/HDL_Forensics

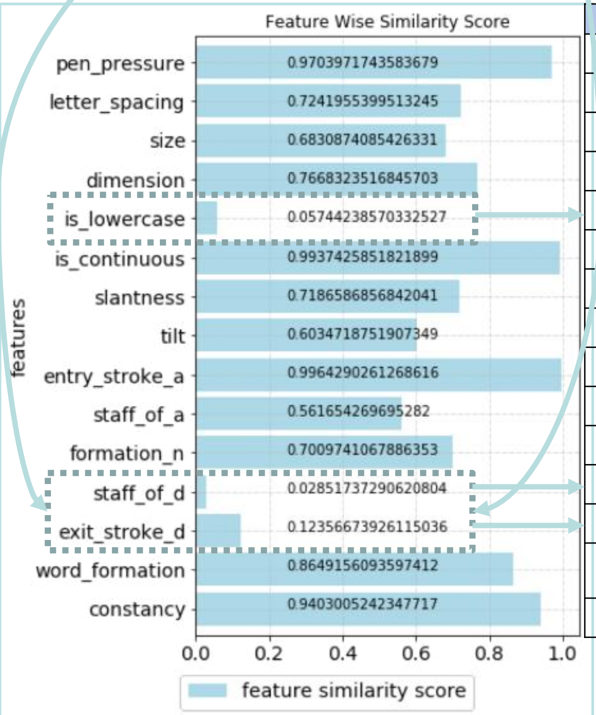
Explanation of Comparison

Similarity Score: 0.3784

0103b_num1.png 0012c_num3.png



Cosine Similarity Threshold: **0.8**
True Negative



Left Classes	Right Classes
Strong	Strong
High	Medium
Large	Medium
Medium	Medium
No	Yes
No	No
Normal	Normal
Normal	Tilted
No Stroke	No Stroke
No Staff	Tented
Normal	No Formation
No Staff	Loopy
No Stroke	Downstroke
Not Wellformed	Not Wellformed
Irregular	Irregular

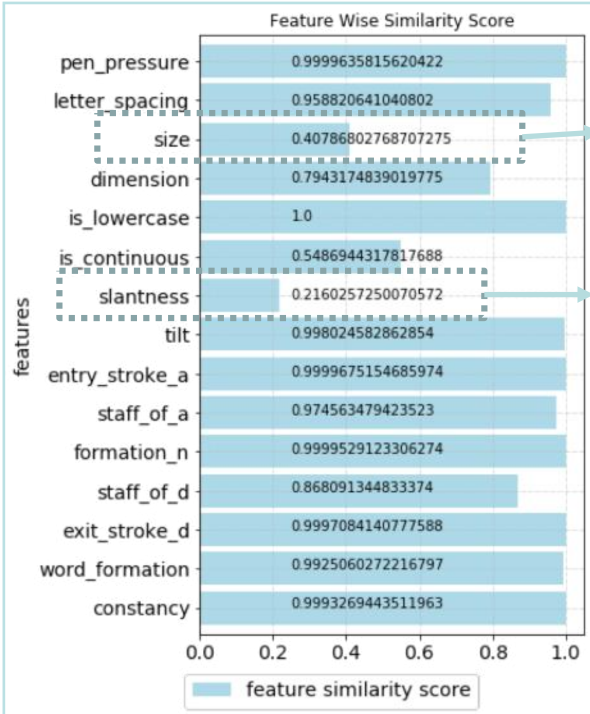
Table 5: Predicted Classes

Image 6: Graph of Similarity Score of Left & Right Image Features
Different Writer Samples

Similarity Score: 0.7291

0021c_num8.png 0021a_num1.png

Cosine Similarity Threshold: **0.8**
False Negative



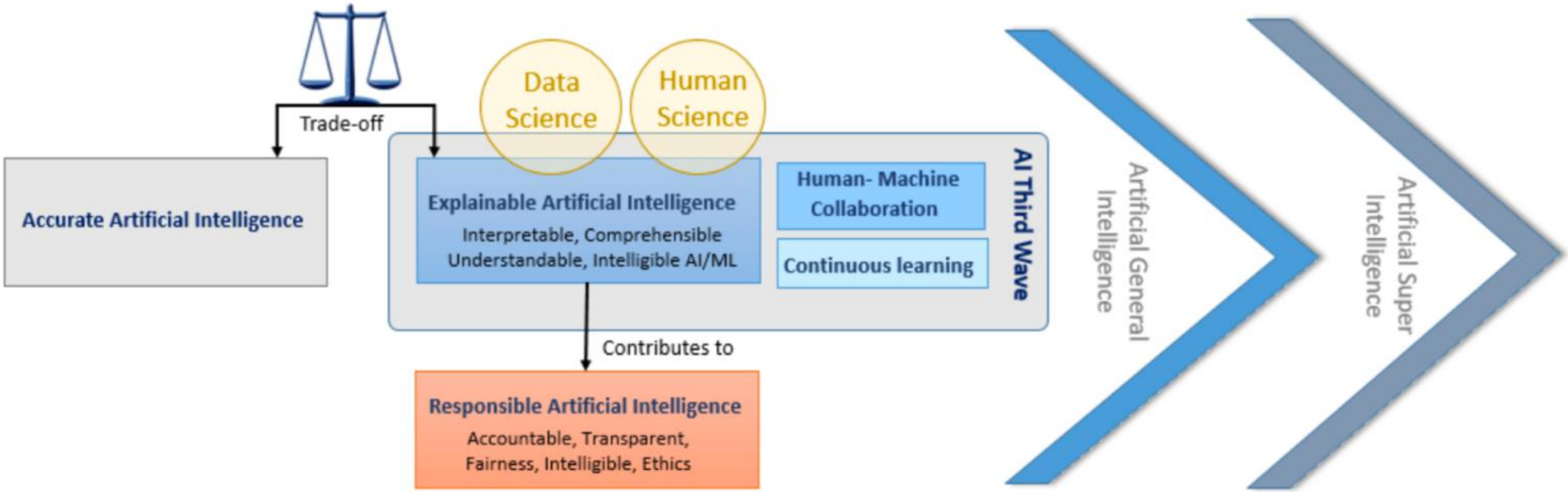
Left Classes	Right Classes
Medium	Medium
Medium	Medium
Large	Medium
Medium	Medium
Yes	Yes
No	Yes
Normal	Slight Right
Normal	Normal
No Stroke	No Stroke
Retraced	Retraced
Normal	Normal
Loopy	Loopy
Downstroke	Downstroke
Wellformed	Wellformed
Regular	Regular

Table 5: Predicted Classes

Image 7: Graph of Similarity Score of Left & Right Image Features
Same Writer Samples

Report Card of Comparing two Samples

Future of XAI



Source: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8466590

Summary and Conclusion

- Deep Learning models are often blackboxes
- Ante-hoc XAI models are new learning models with an explanatory interface
- Post-hoc models retain high performance
- Several new ideas developing: Bayesian Teaching, Causal learning of deep neural nets
- Probabilistic AI is useful for deep learning
- XAI may be an inflection point in AI