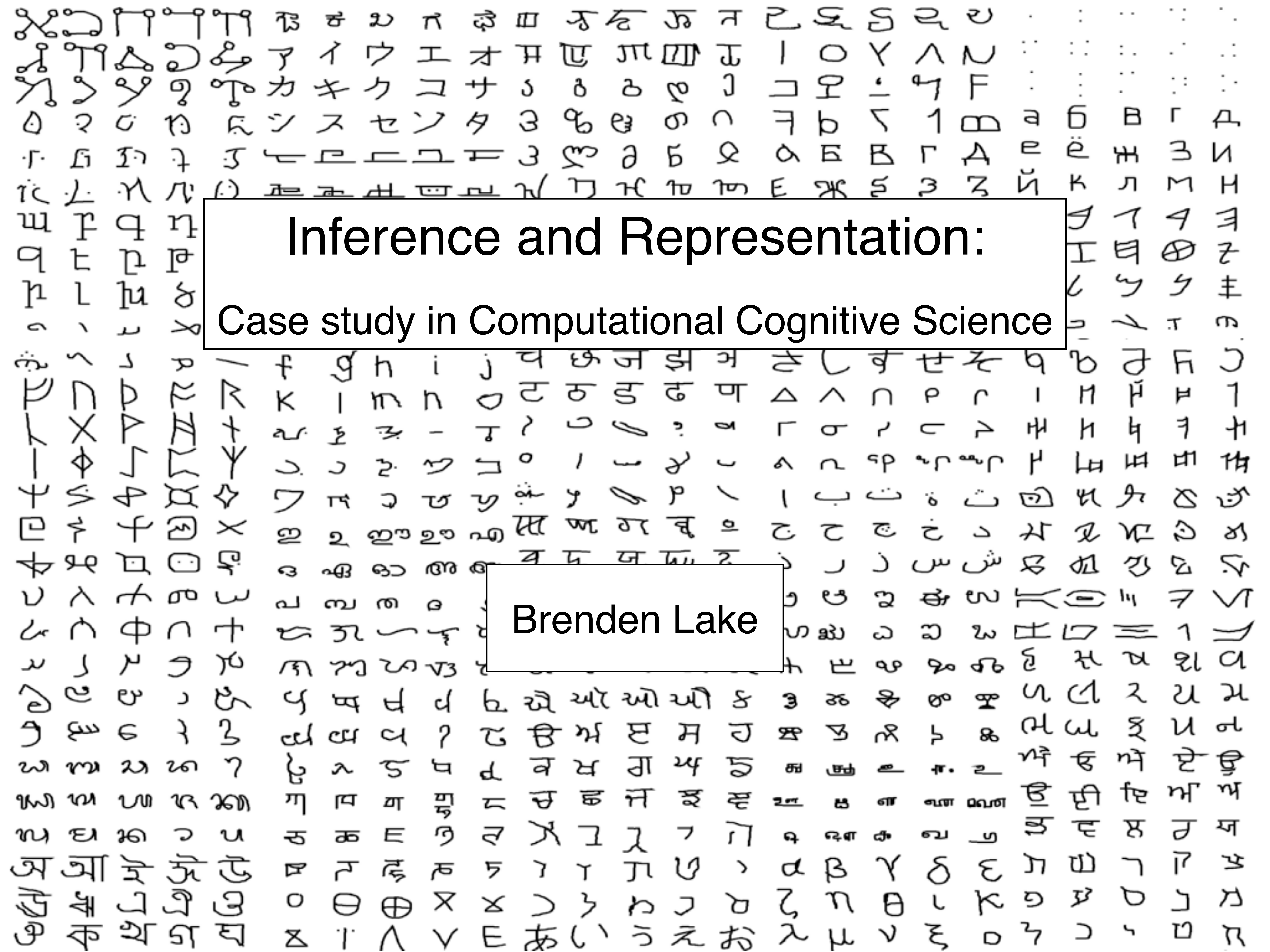


Inference and Representation:

Case study in Computational Cognitive Science

Brenden Lake

“Learning classifiers” in cognitive science

concept learning
(cognitive science
&
psychology)

=

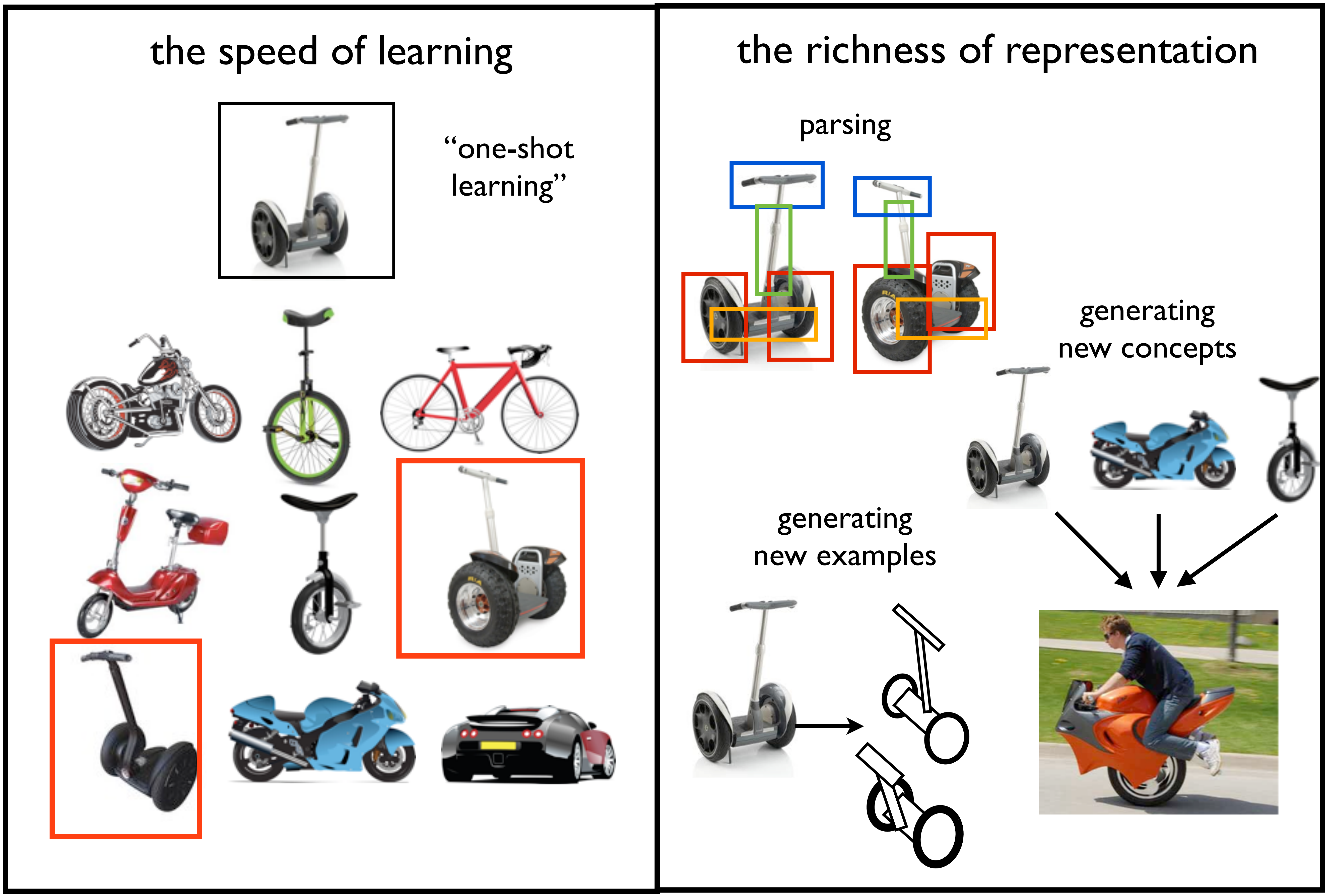
classification
(data science
&
machine learning)

labeled data for
“dogs”

generalization task:
dog or cat?

labeled data for
“cats”

human-level concept learning



portable immersion circulator

bucket-wheel excavator

spring-loaded camming device

drawing knife

A testbed for studying human-level concept learning

We would like to investigate a domain with…

1) A relatively even slate for comparing humans and machines.

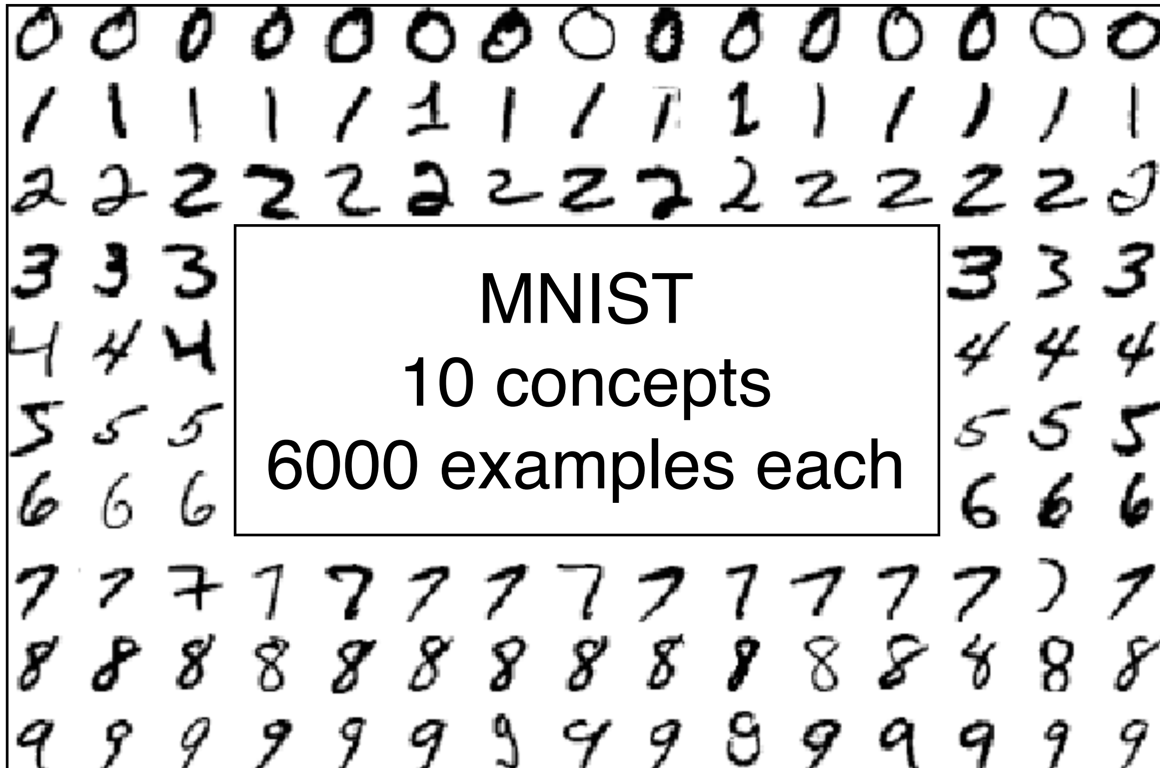
2) Natural, high-dimensional concepts.

3) A reasonable chance of building computational models that can see most of the structure that people see.

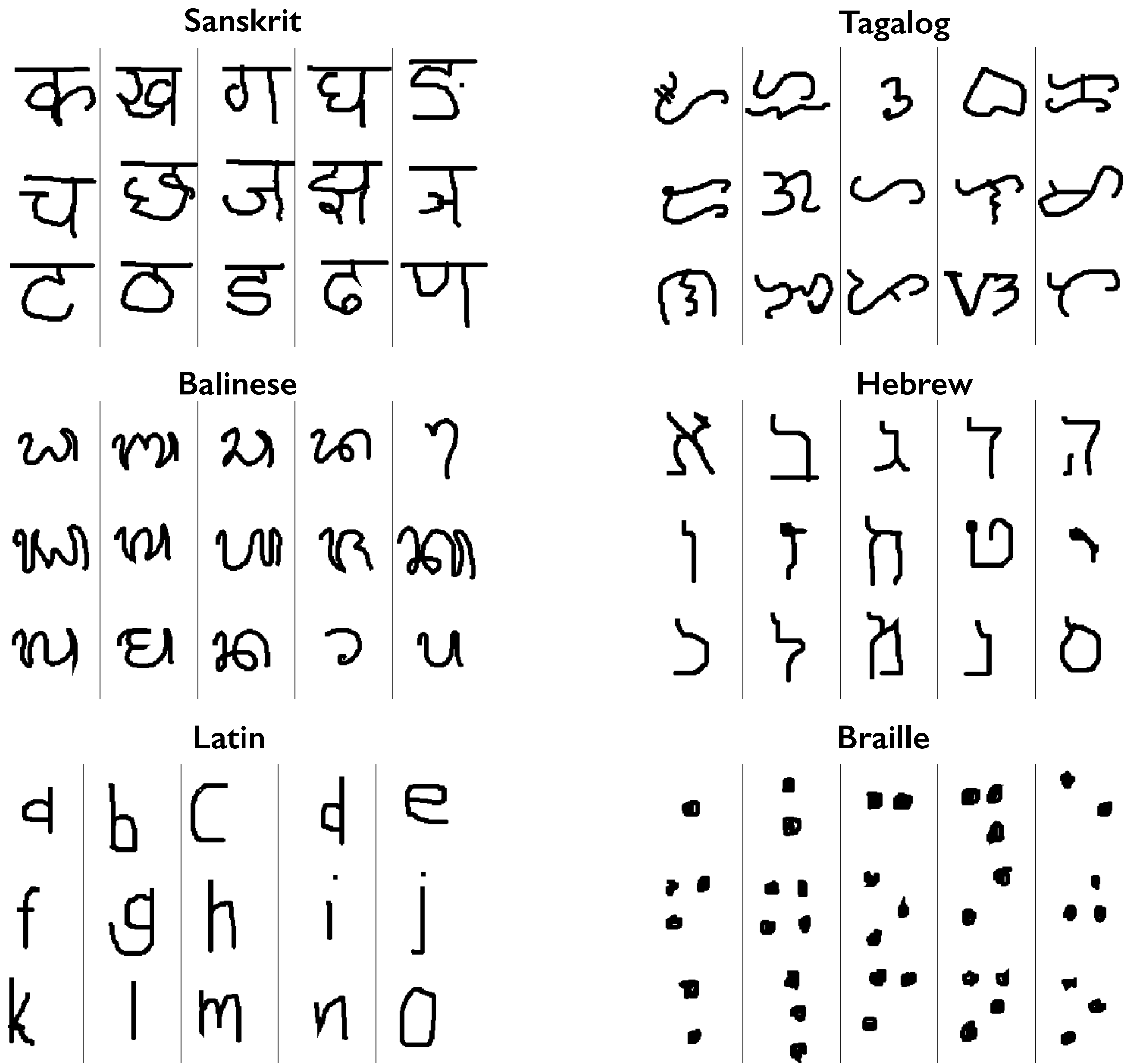
4) Insights that generalize across domains.

Standard machine learning

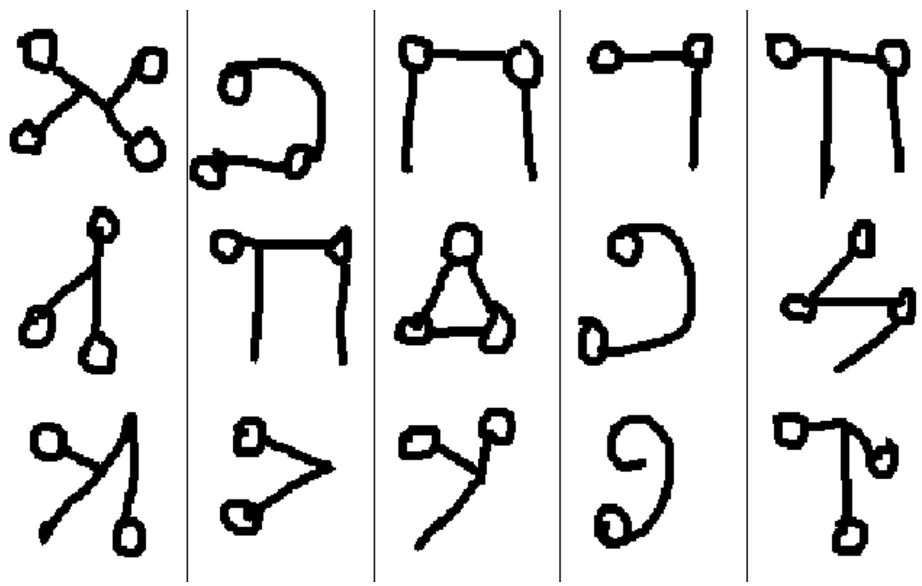
https://github.com/brendenlake



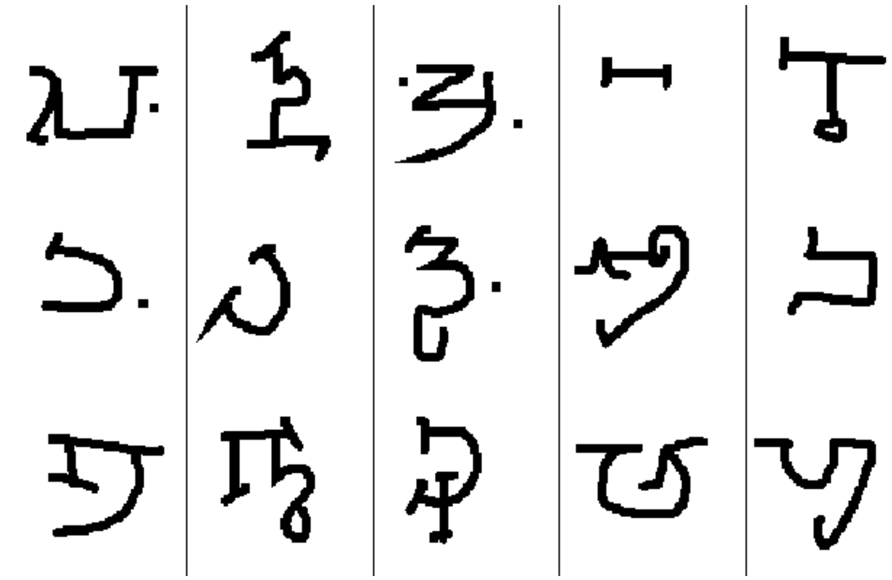
Our testbed


Sanskrit
Tagalog
Balinese
Hebrew
Latin
Braille

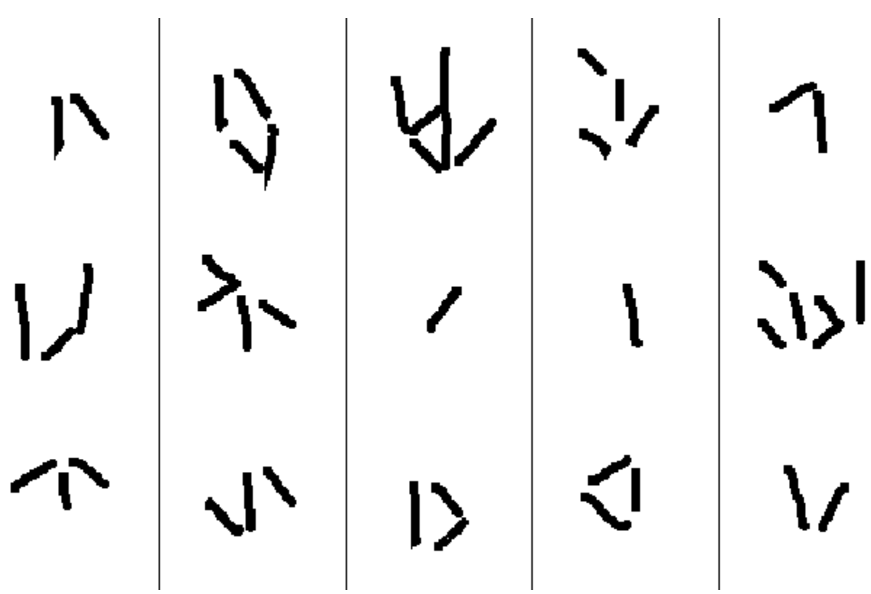
Angelic



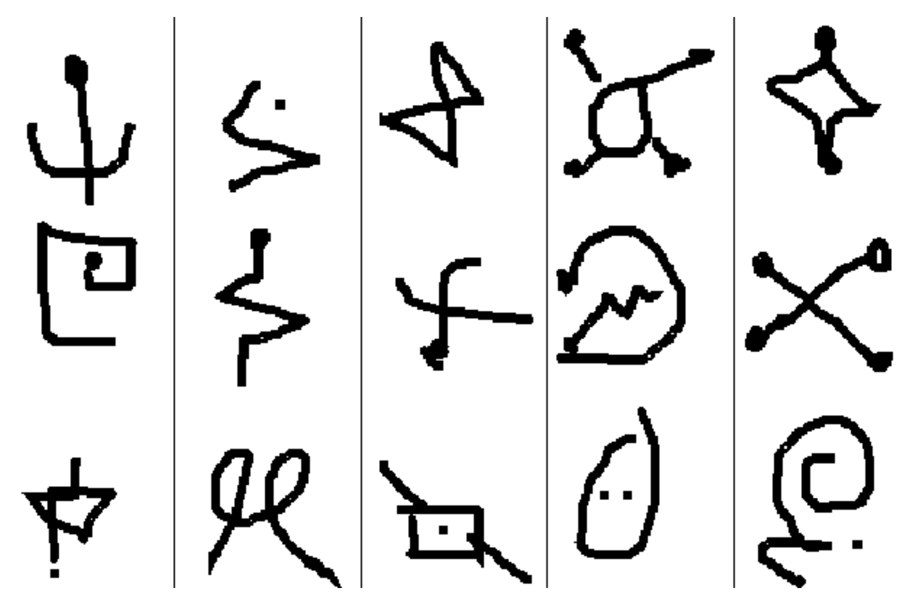
Alphabet of the Magi



ULOG

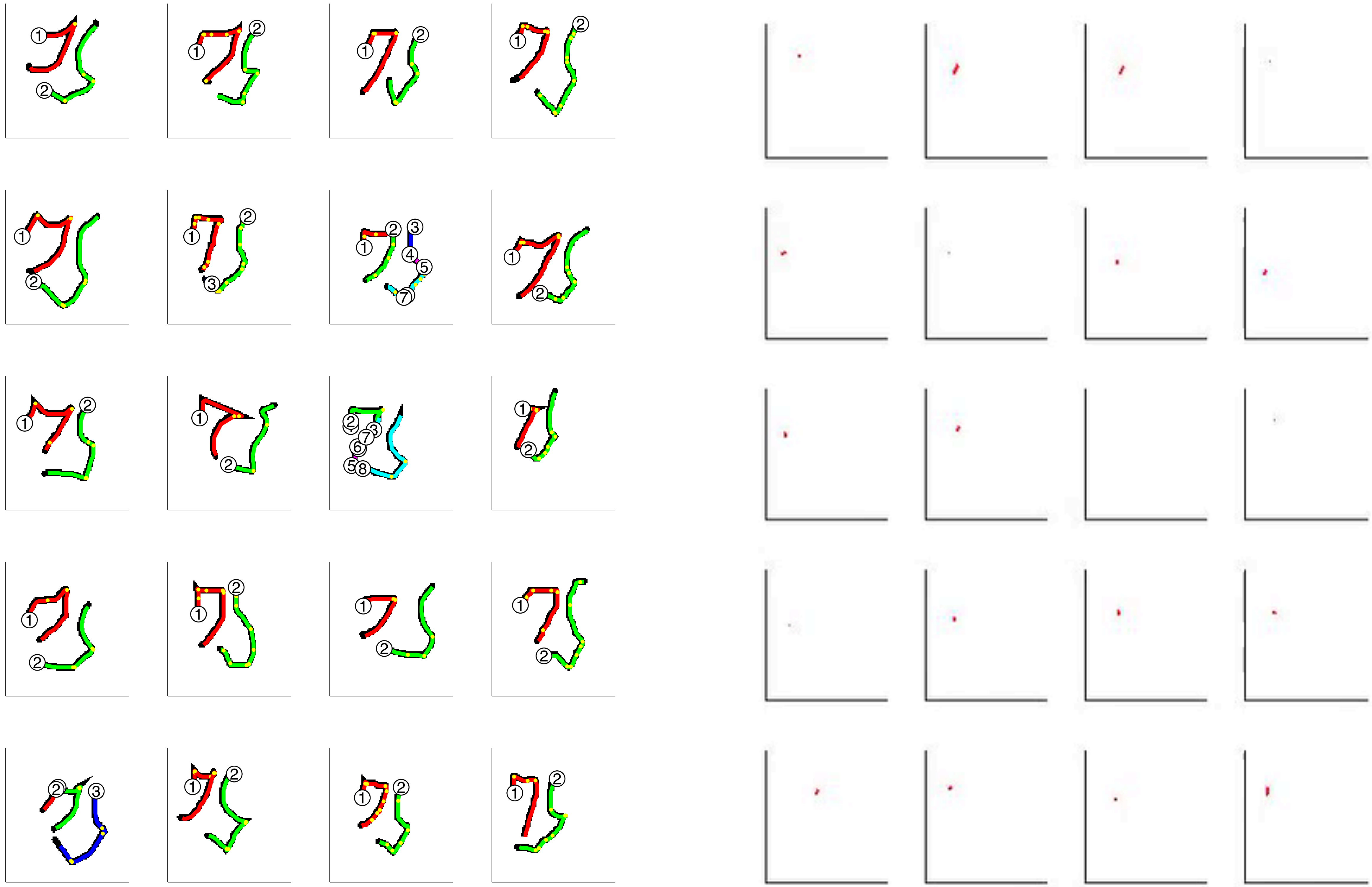


Futurama



Original Image

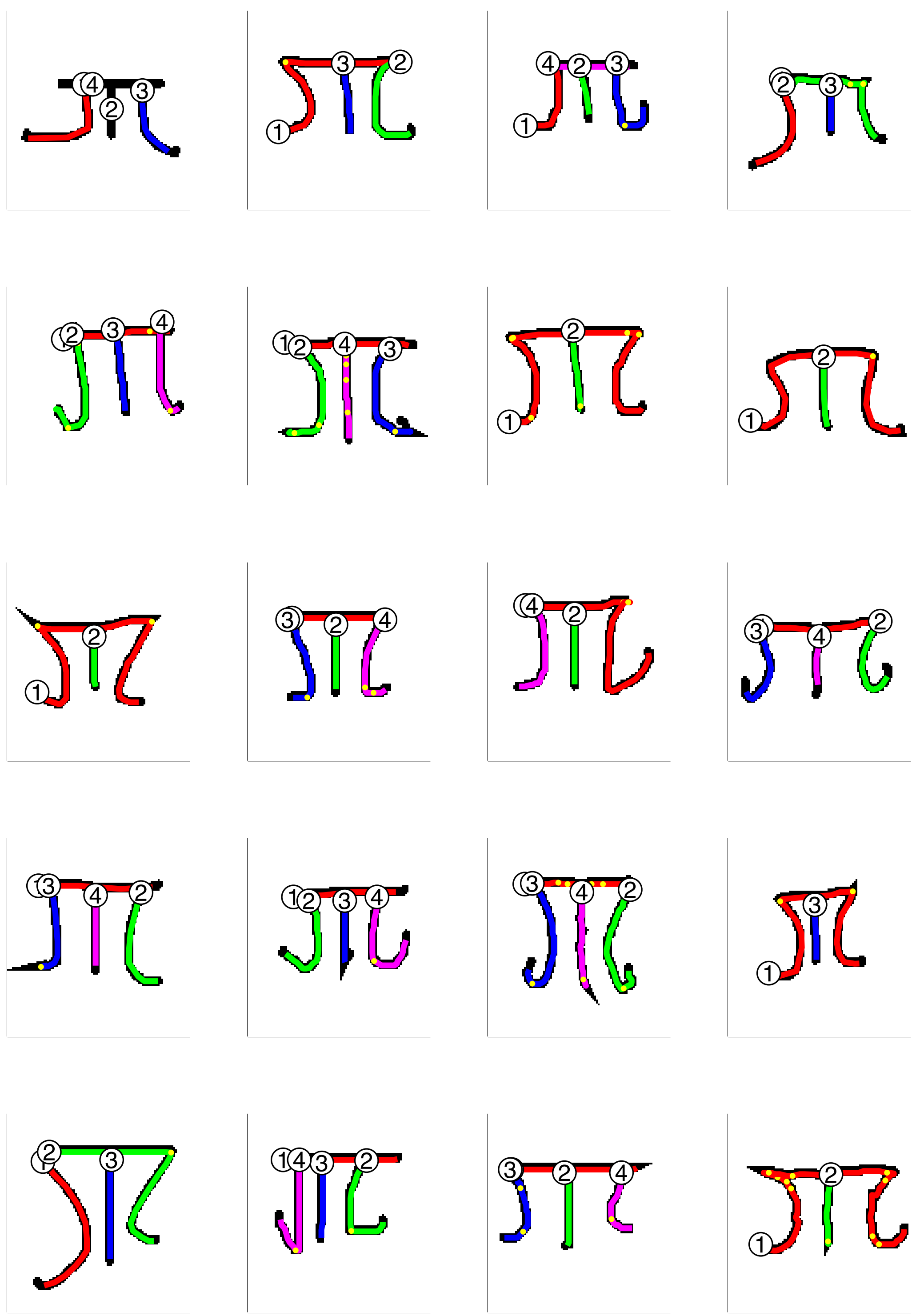
20 People's Strokes

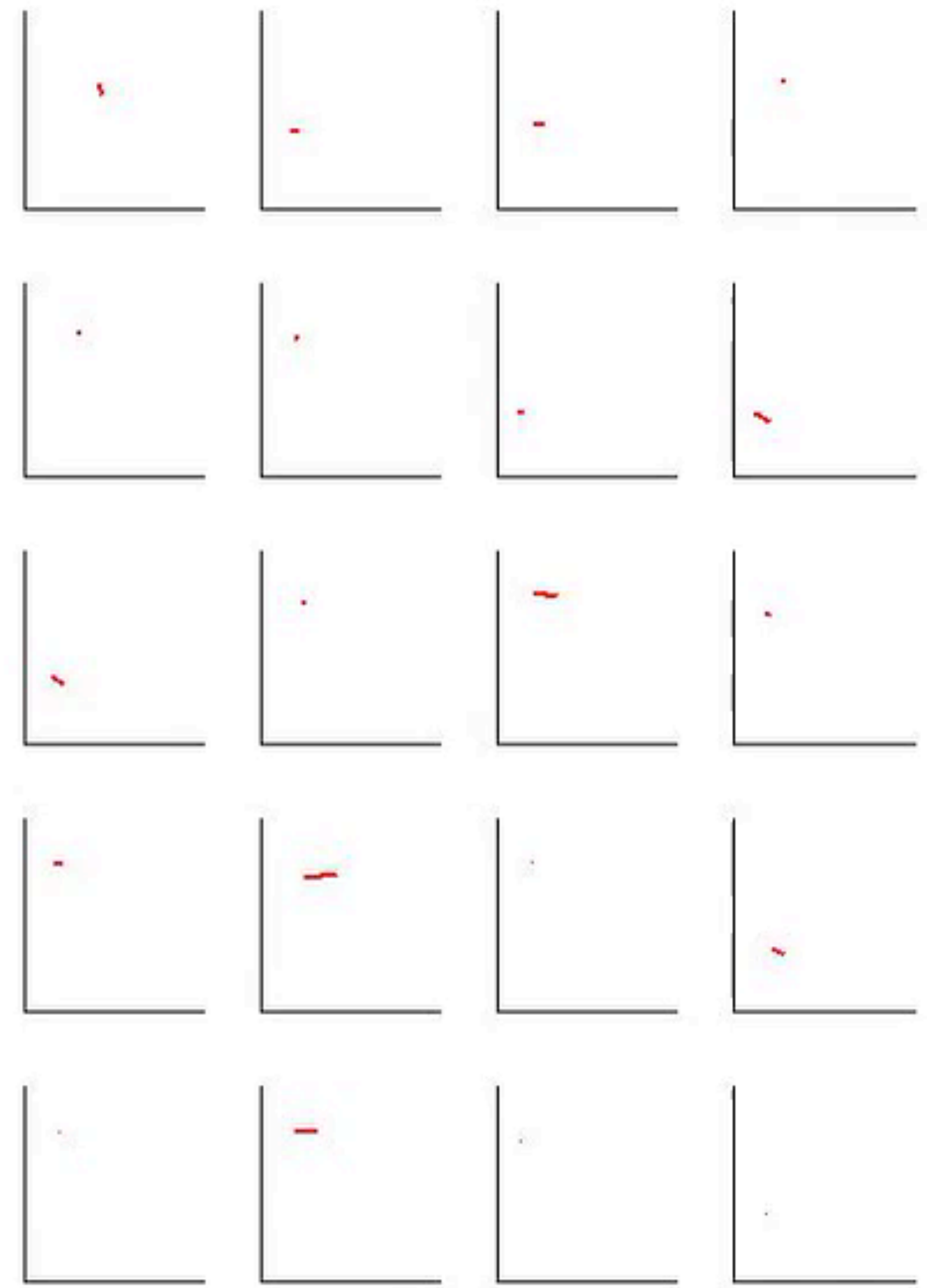


Stroke order:

Original Image

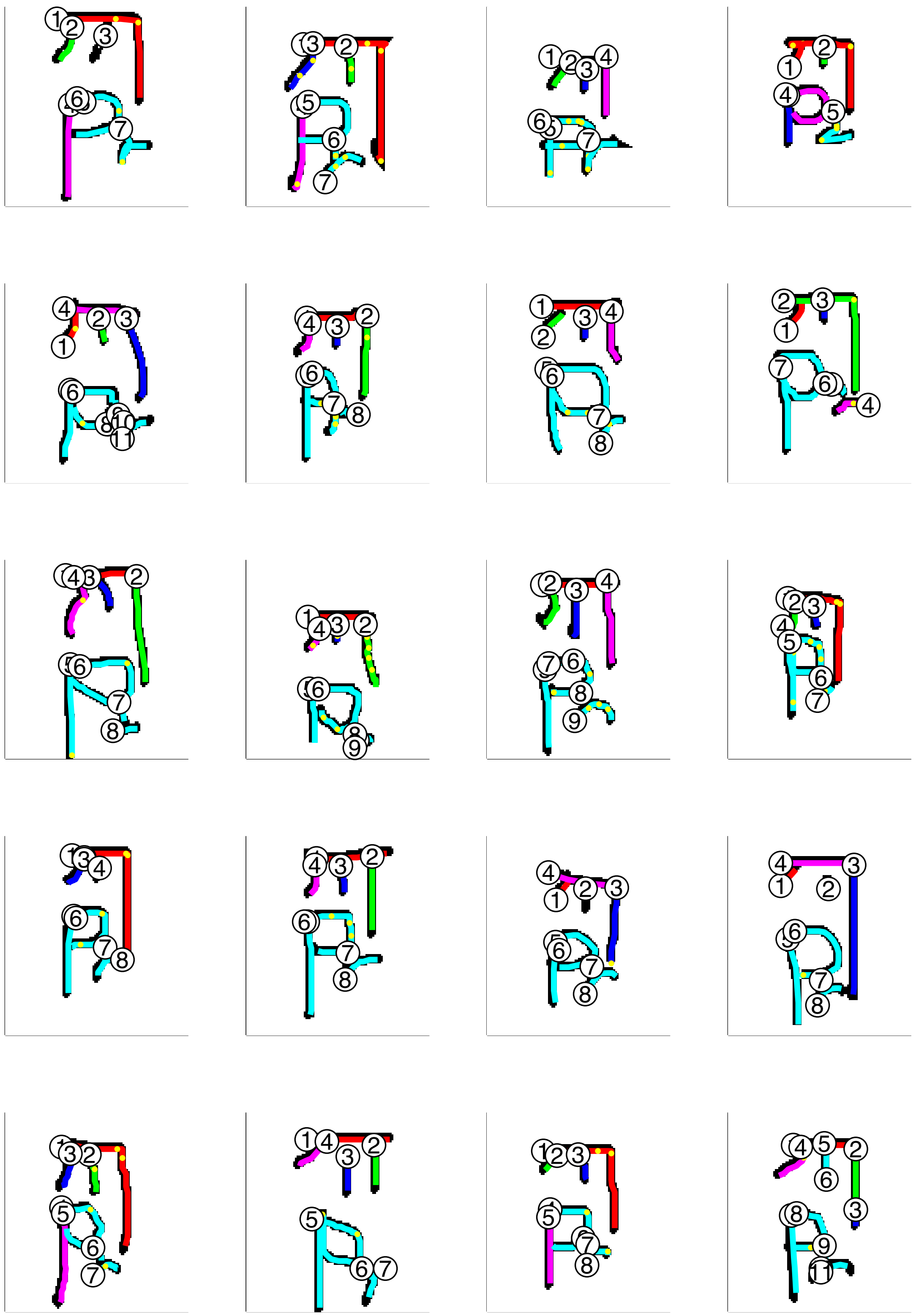
20 People's Strokes

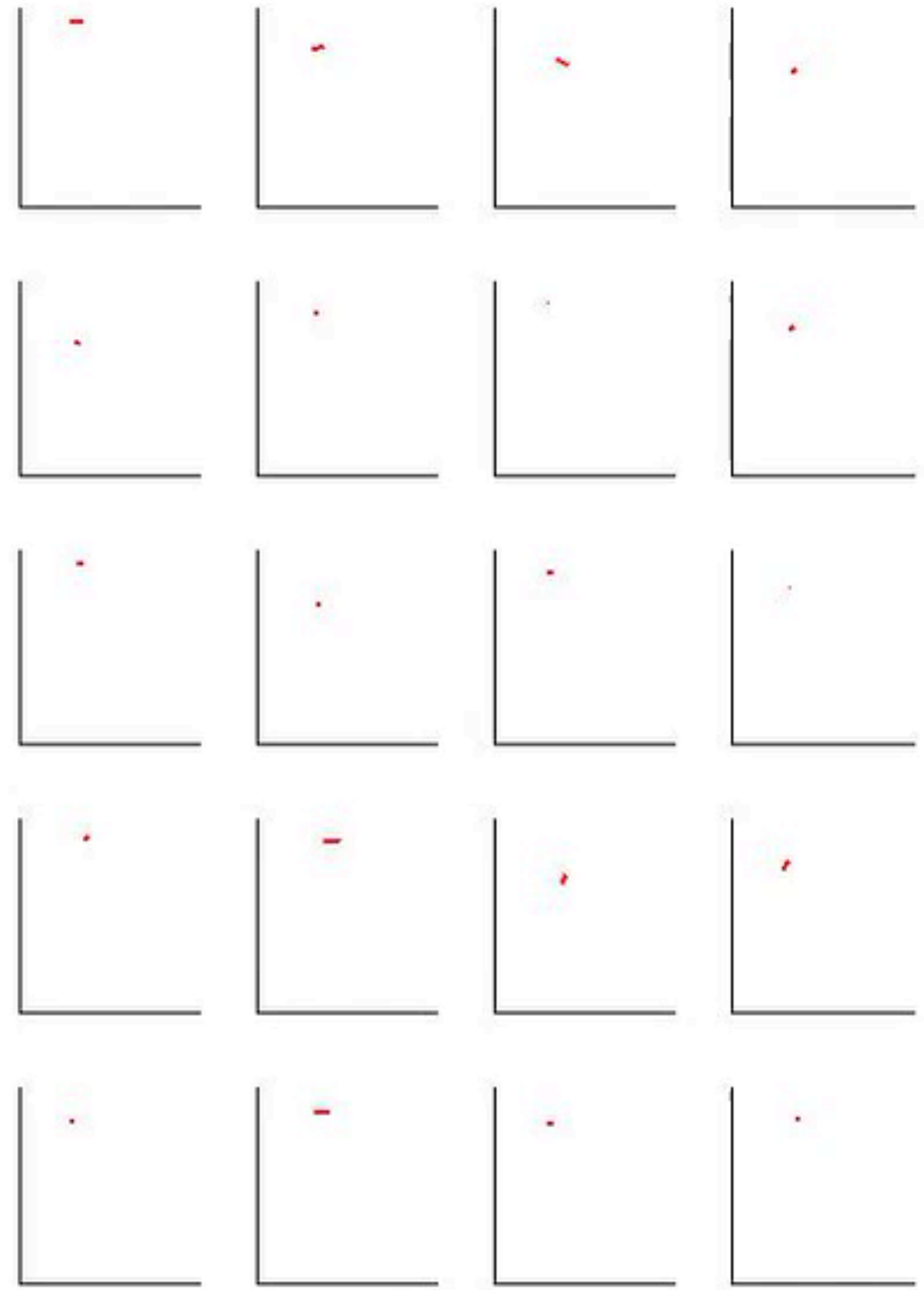




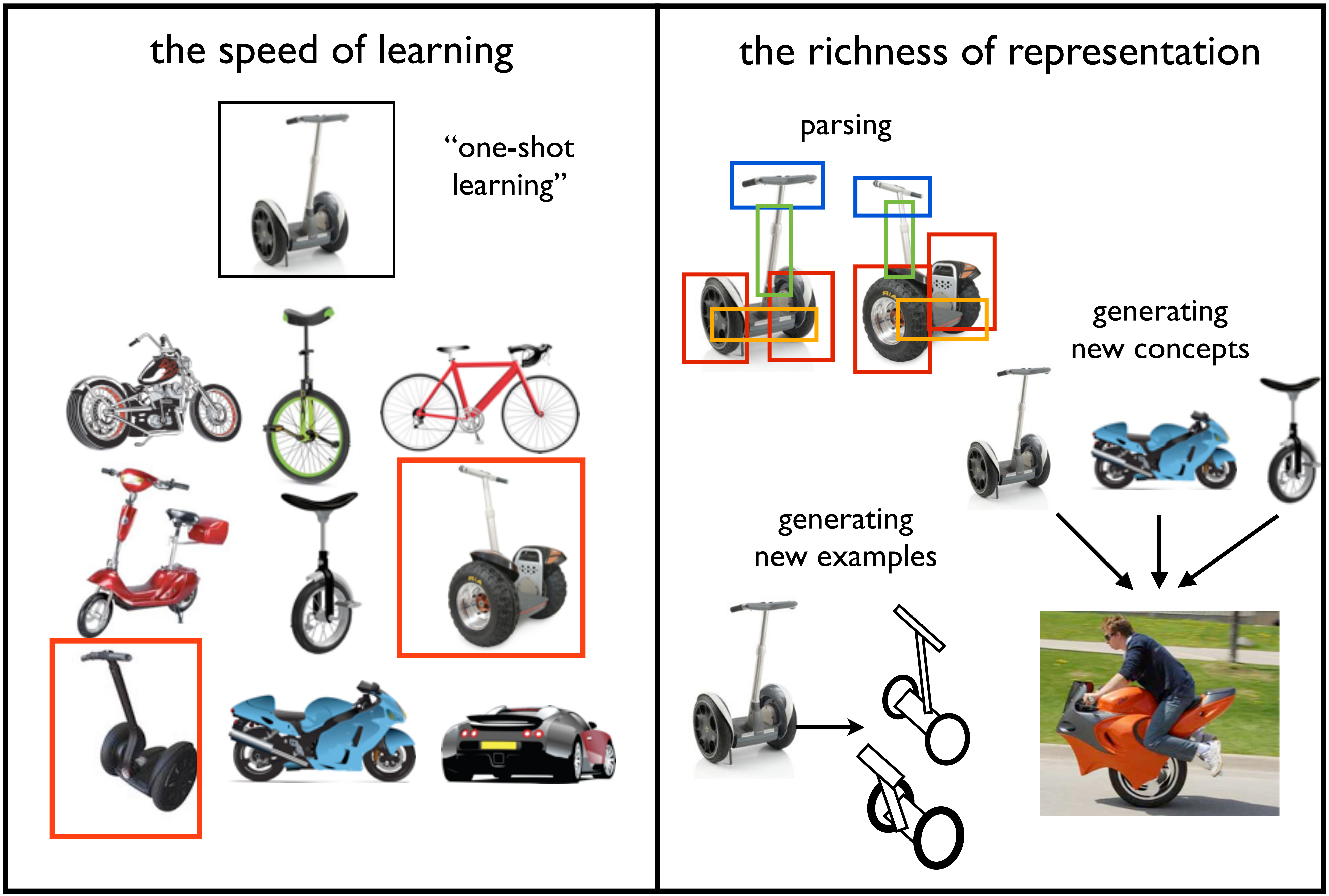
Original Image

20 People's Strokes



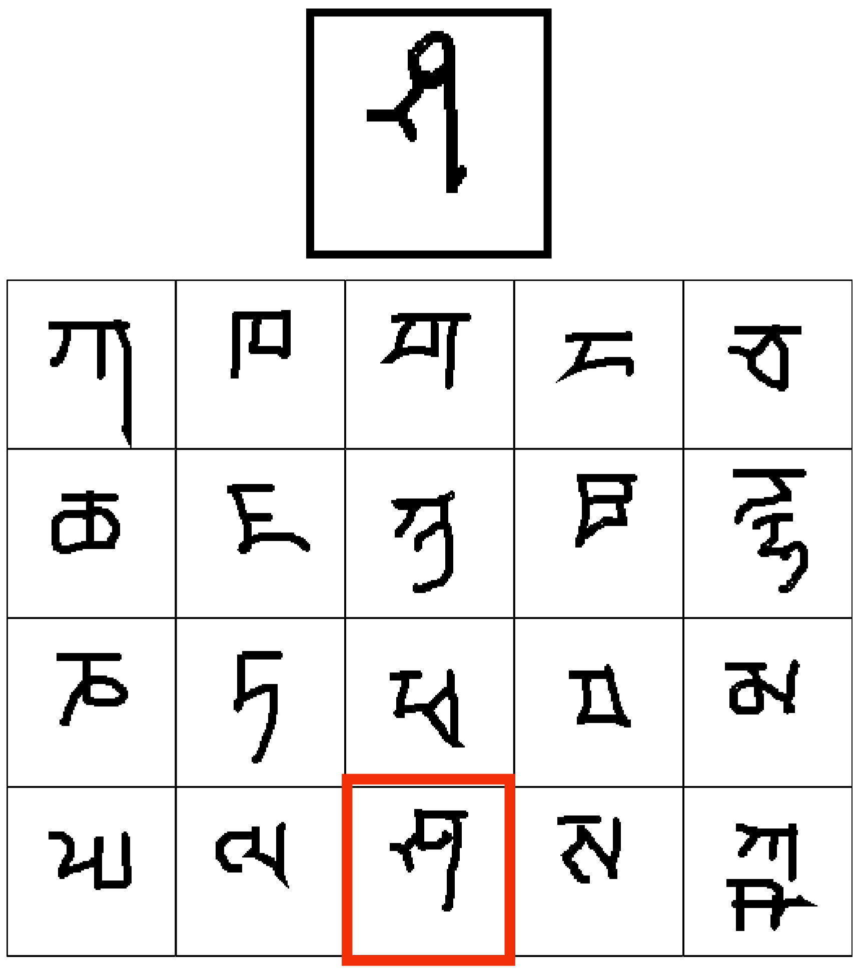


human-level concept learning



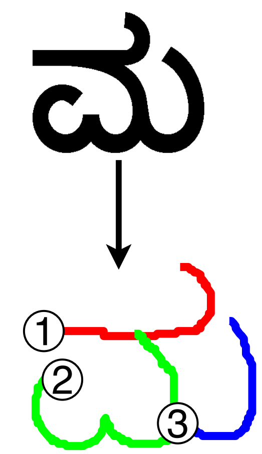
human-level concept learning

the speed of learning

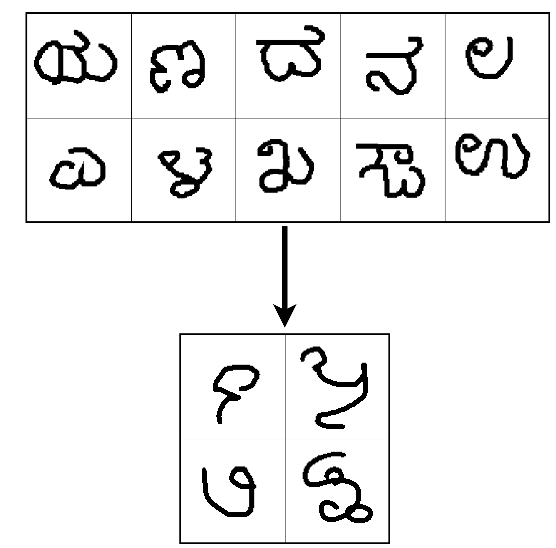


the richness of representation

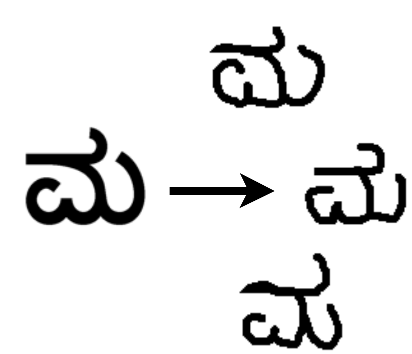
parsing



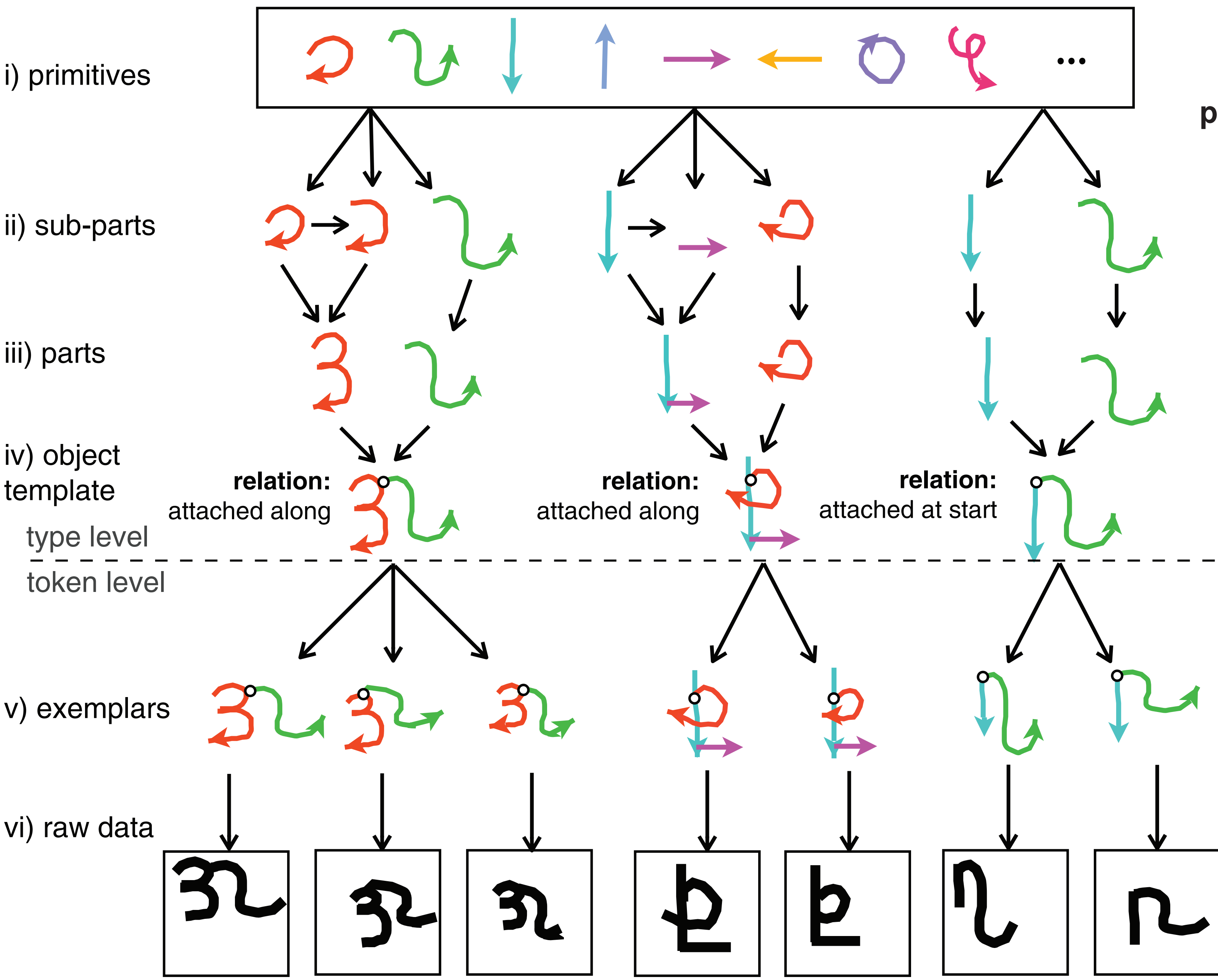
generating
new concepts



generating
new examples



Bayesian Program Learning

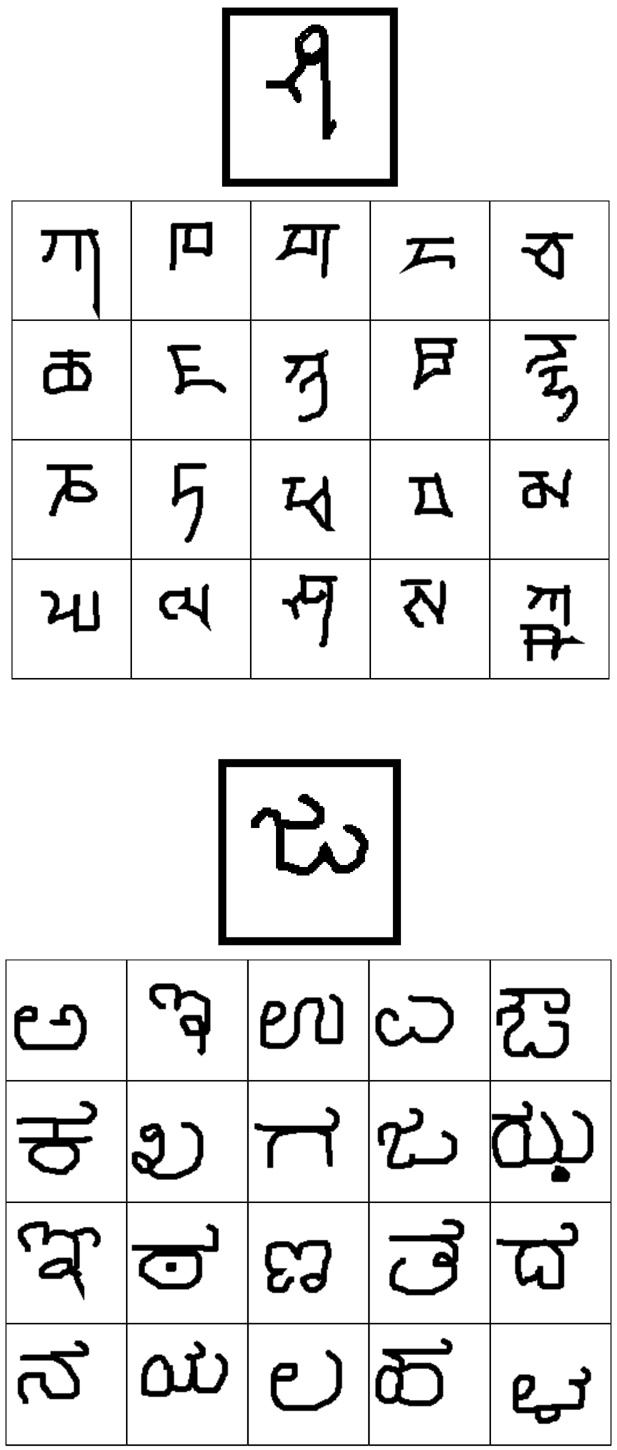


procedure GENERATETYPE
  $\kappa \leftarrow P(\kappa)$ ▷ Sample number of parts
  for i = 1 ... κ **do**
    $n_i \leftarrow P(n_i|\kappa)$ ▷ Sample number of sub-parts
    for j = 1 ... n_i **do**
      $s_{ij} \leftarrow P(s_{ij}|s_{i(j-1)})$ ▷ Sample sub-part sequence
    end for
    $R_i \leftarrow P(R_i|S_1, ..., S_{i-1})$ ▷ Sample relation
  end for
  $\psi \leftarrow \{\kappa, R, S\}$
  return @GENERATETOKEN(ψ) ▷ Return program

procedure GENERATETOKEN(ψ)
  for $i = 1 ... \kappa$ **do**
    $S_i^{(m)} \leftarrow P(S_i^{(m)}|S_i)$ ▷ Add motor variance
    $L_i^{(m)} \leftarrow P(L_i^{(m)}|R_i, T_1^{(m)}, ..., T_{i-1}^{(m)})$
    ▷ Sample part's start location
    $T_i^{(m)} \leftarrow f(L_i^{(m)}, S_i^{(m)})$ ▷ Compose a part's trajectory
  end for
  $A^{(m)} \leftarrow P(A^{(m)})$ ▷ Sample affine transform
  $I^{(m)} \leftarrow P(I^{(m)}|T^{(m)}, A^{(m)})$ ▷ Sample image
  return $I^{(m)}$

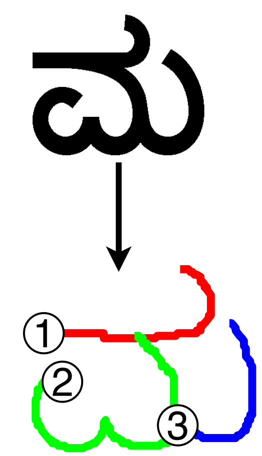
human-level concept learning

the speed of learning

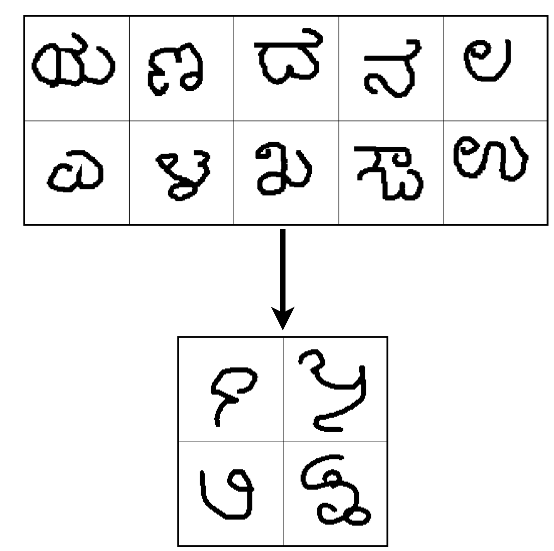


the richness of representation

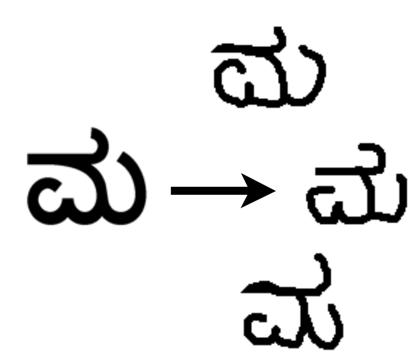
parsing



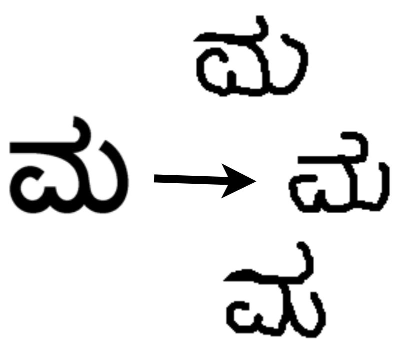
generating
new concepts



generating
new examples

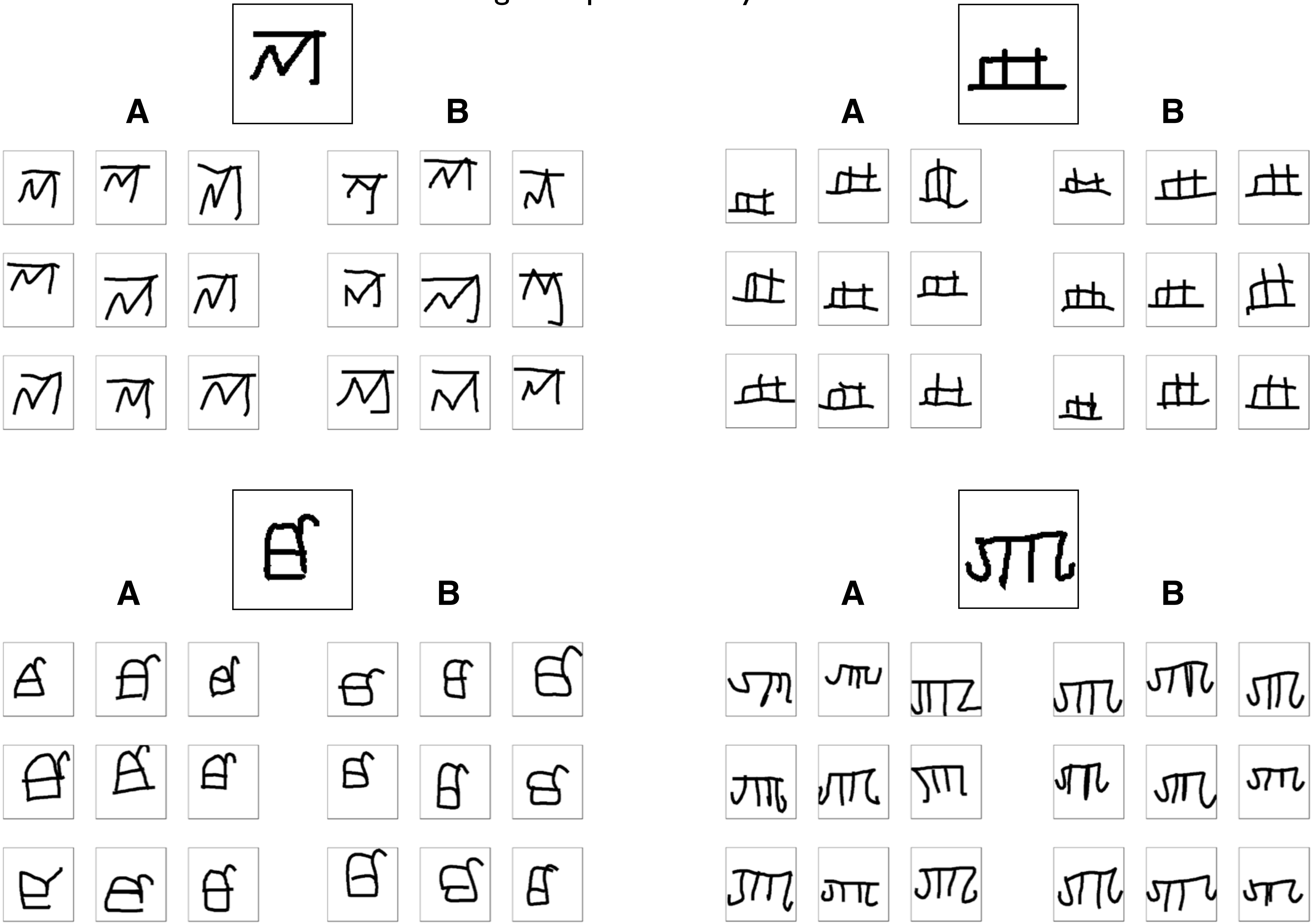


generating new examples



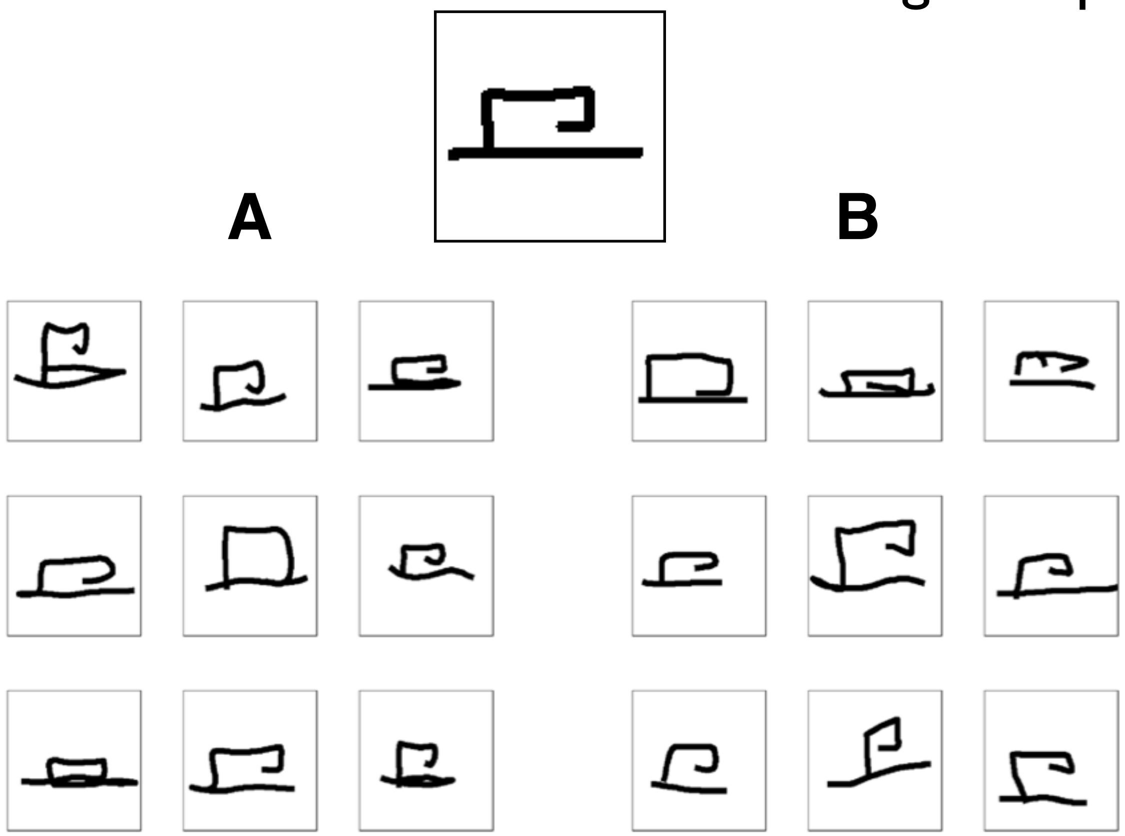
“Draw a new example”

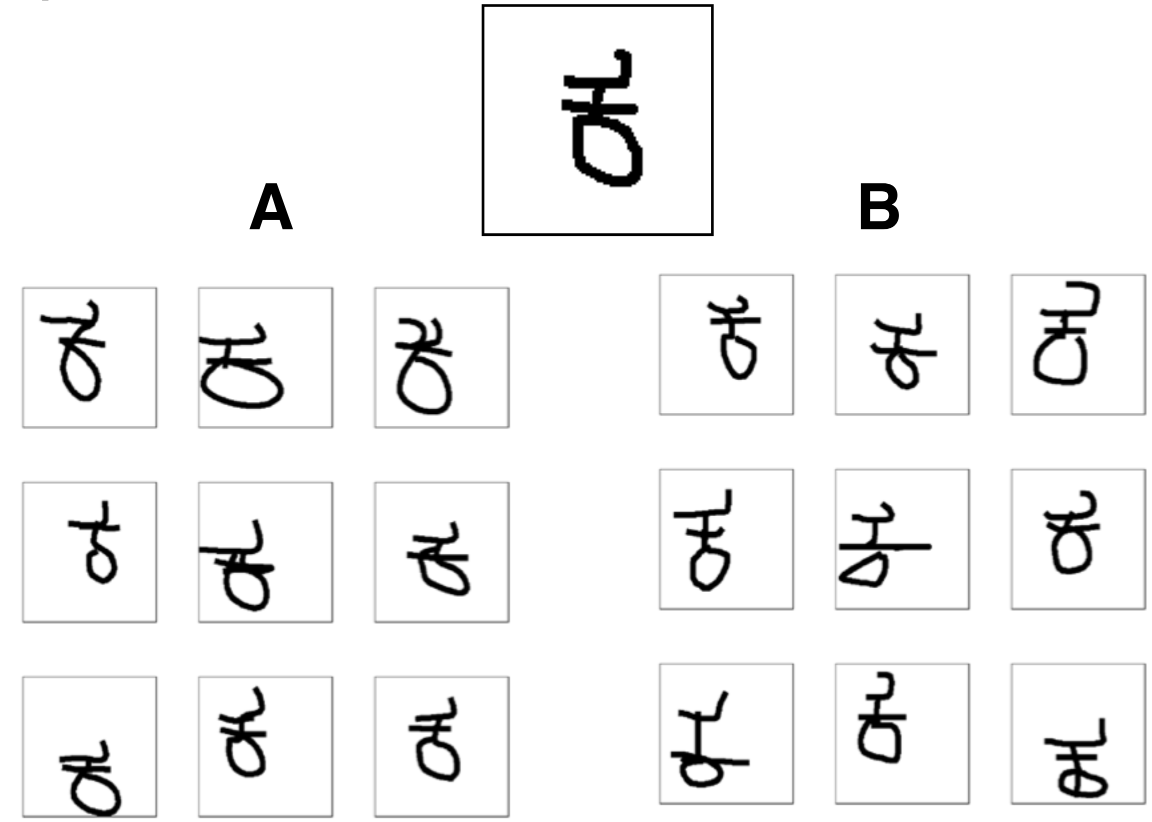
Which grid is produced by the model?

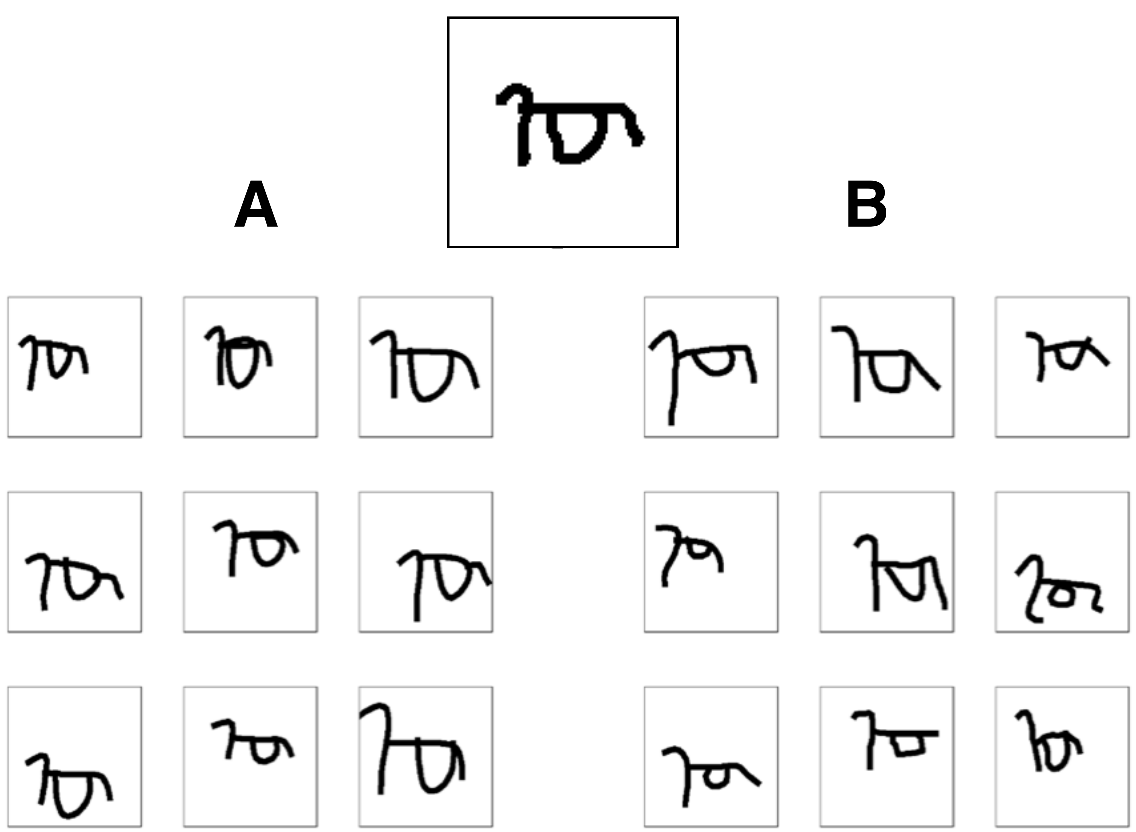


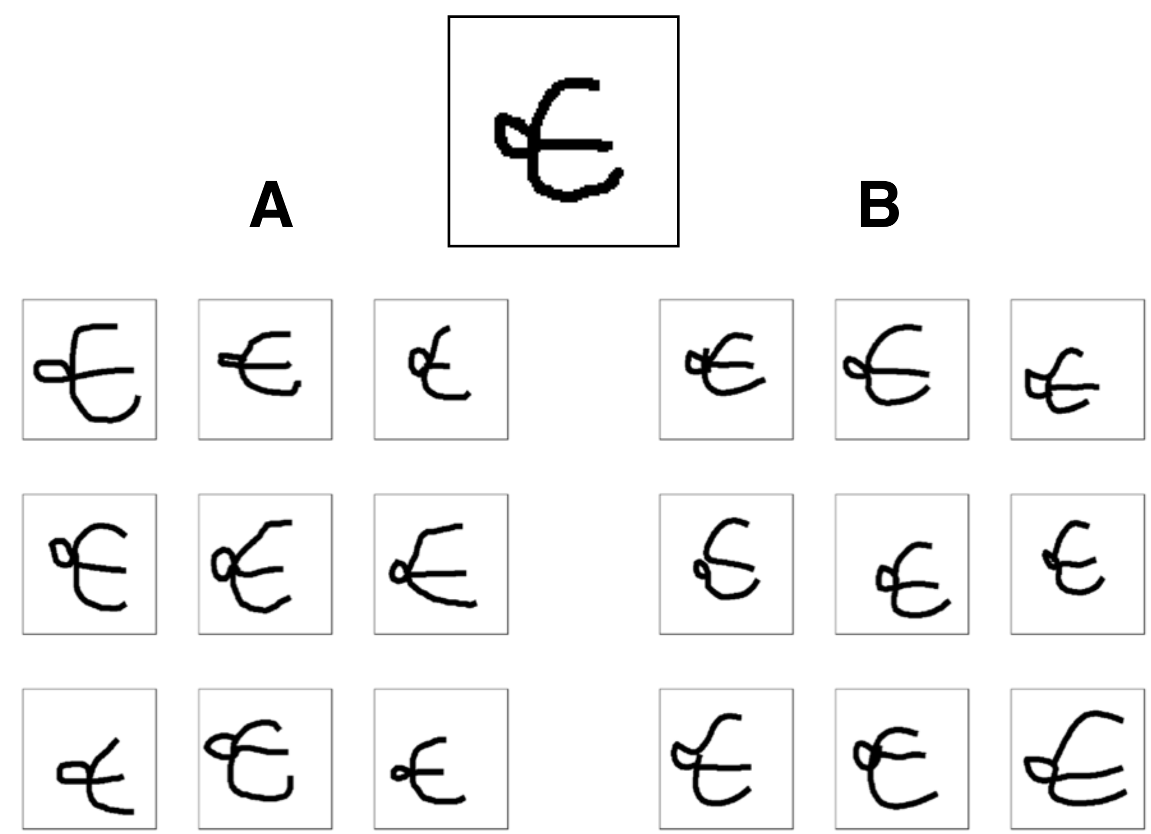
“Draw a new example”

Which grid is produced by the model?



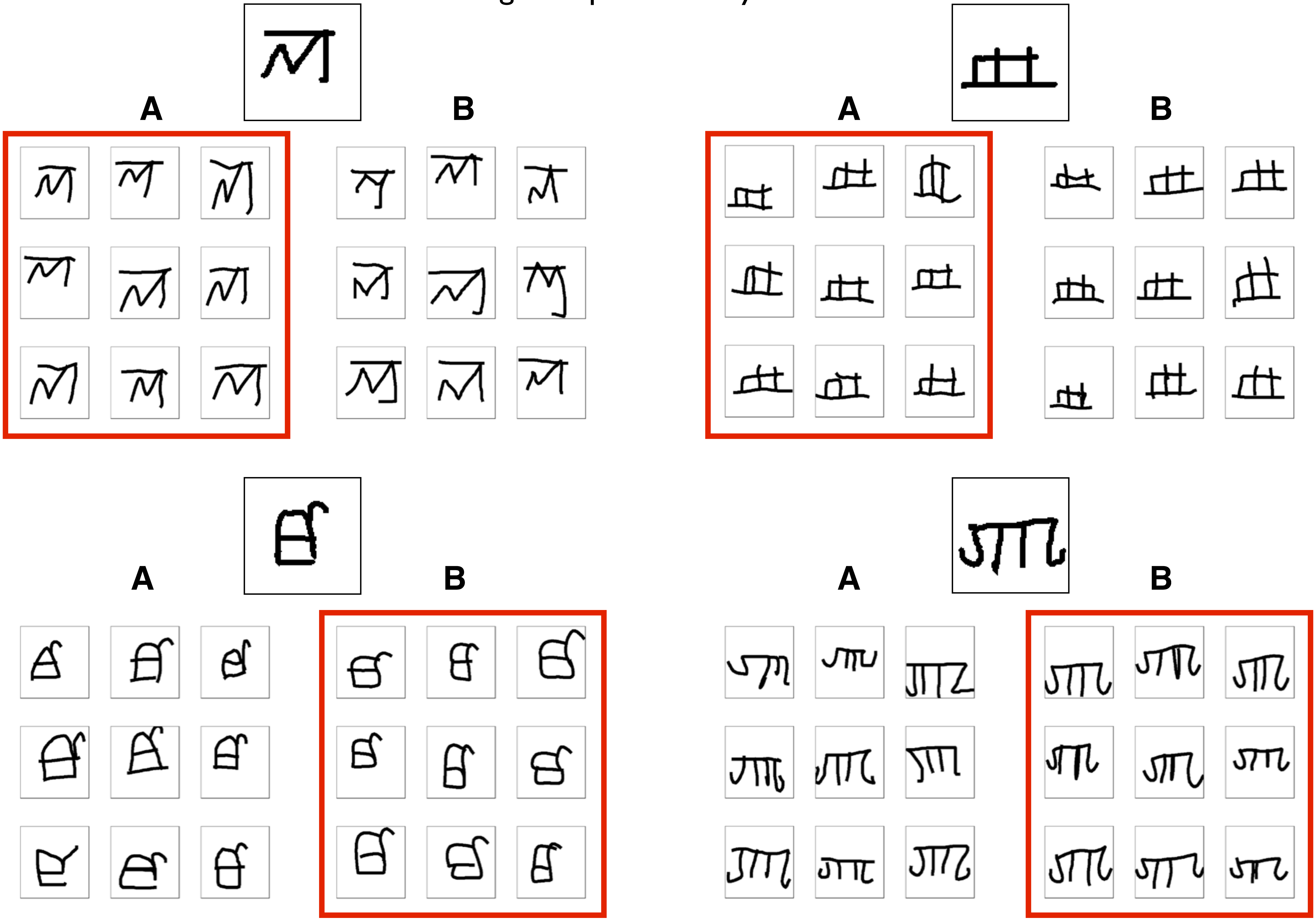






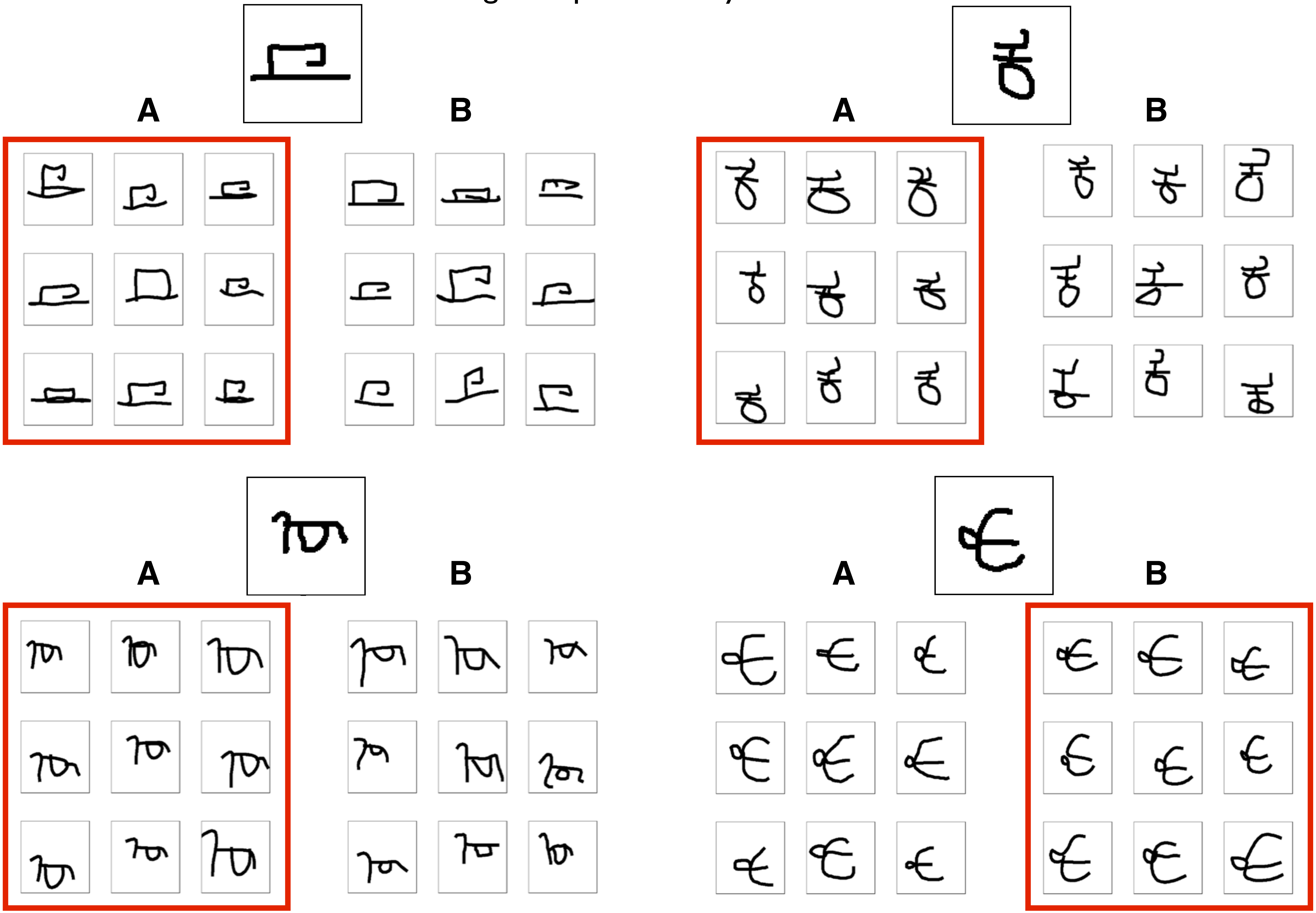
“Draw a new example”

Which grid is produced by the model?



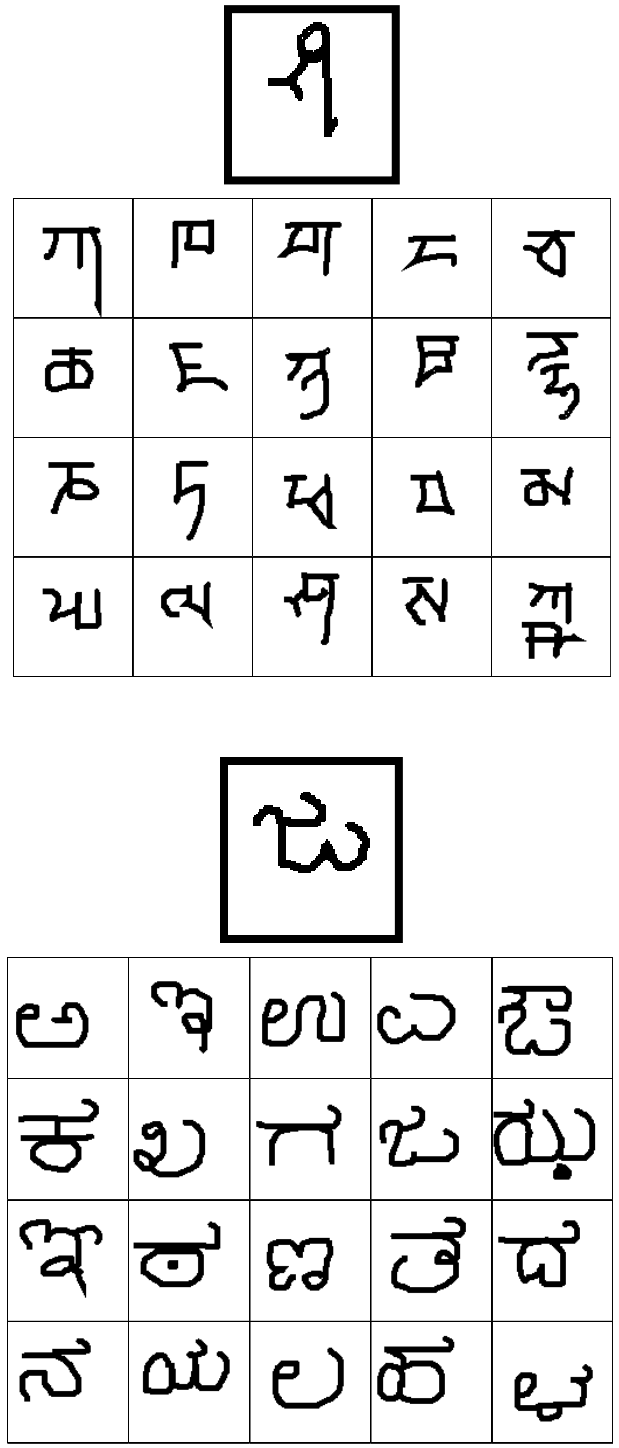
“Draw a new example”

Which grid is produced by the model?



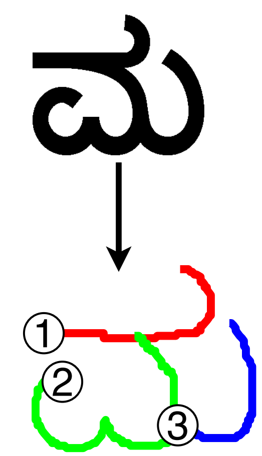
human-level concept learning

the speed of learning

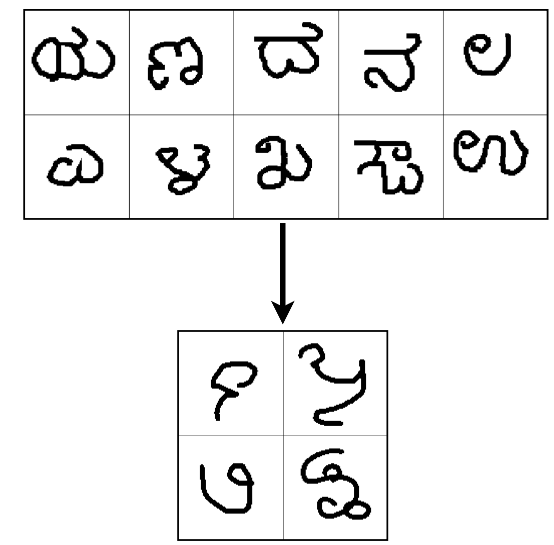


the richness of representation

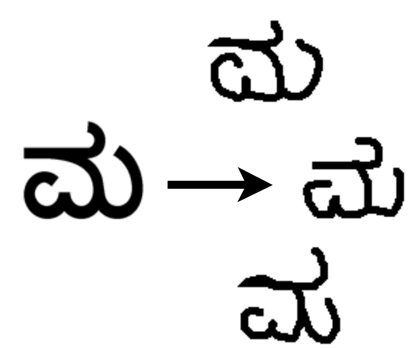
parsing



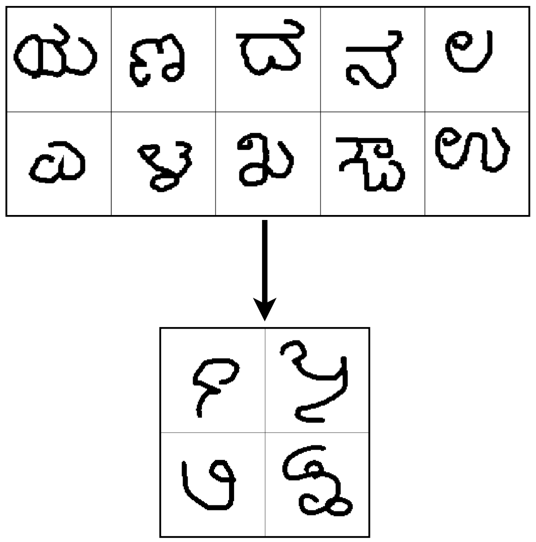
generating
new concepts



generating
new examples



generating
new concepts

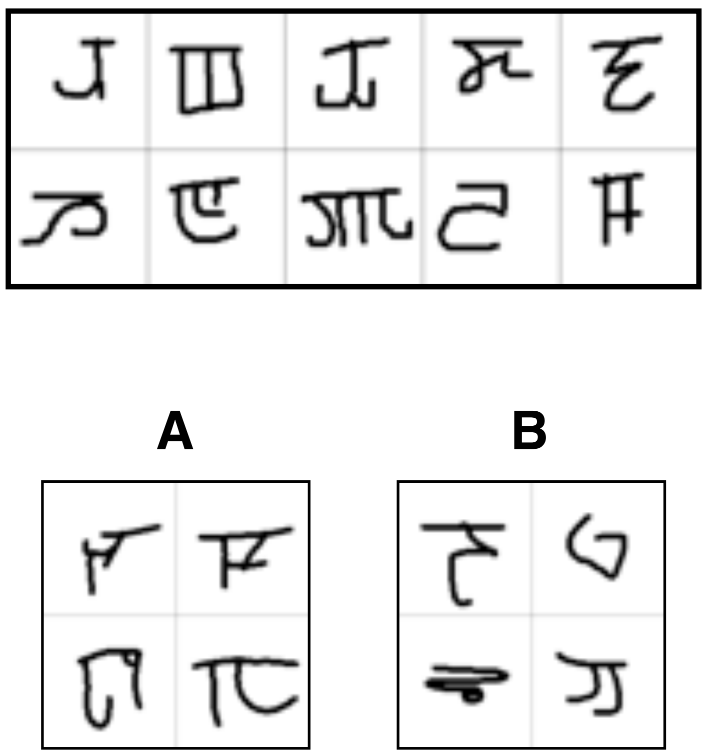


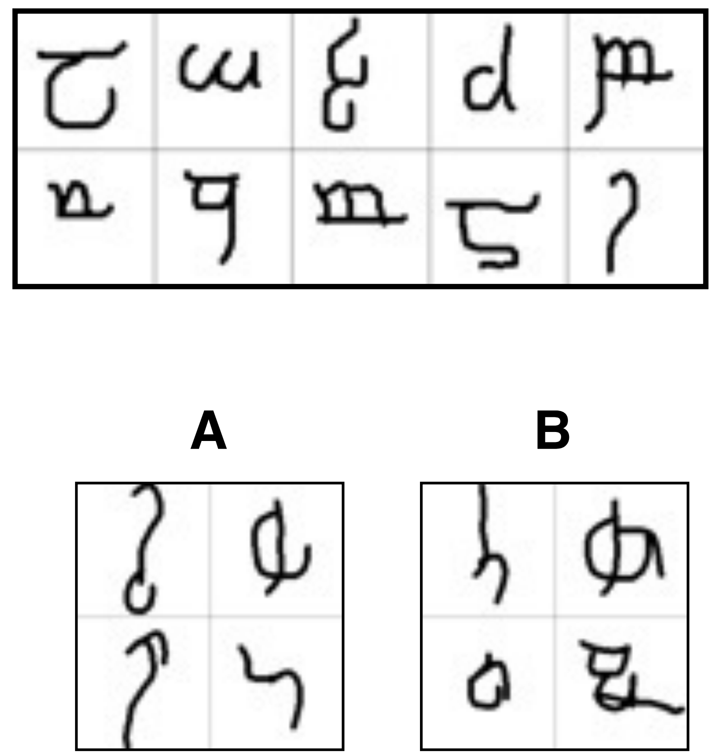
Task: “Design a new character from the same alphabet”

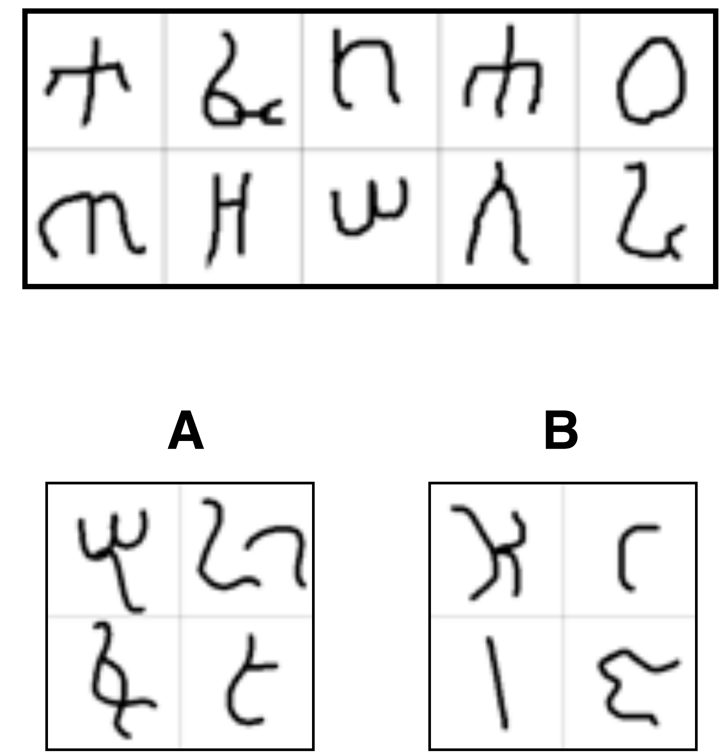
3 seconds
remaining

Task: “Design a new character from the same alphabet”

Which grid is produced by the model?

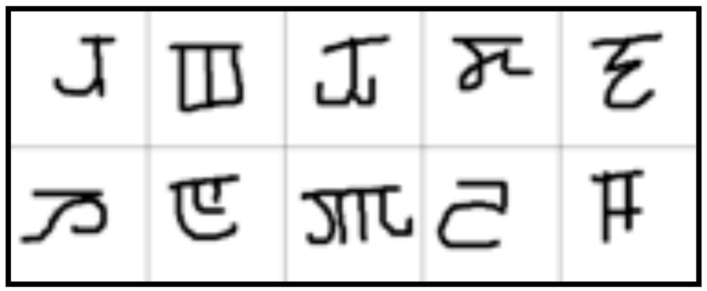


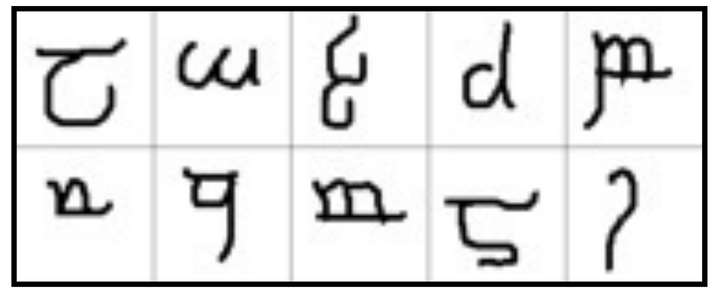


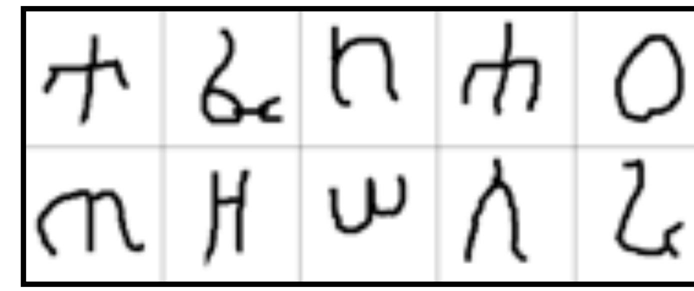


Task: “Design a new character from the same alphabet”

Which grid is produced by the model?

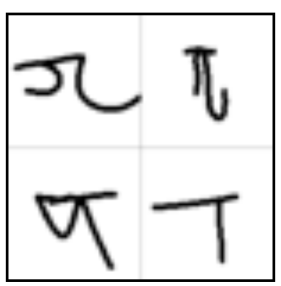






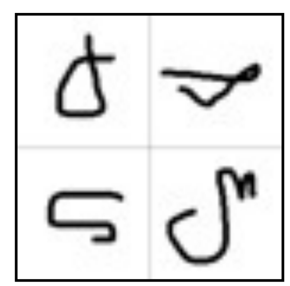
A

B

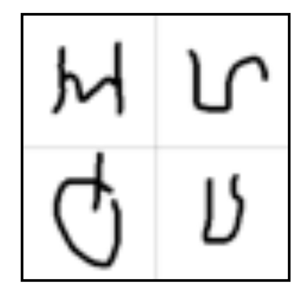


A

B



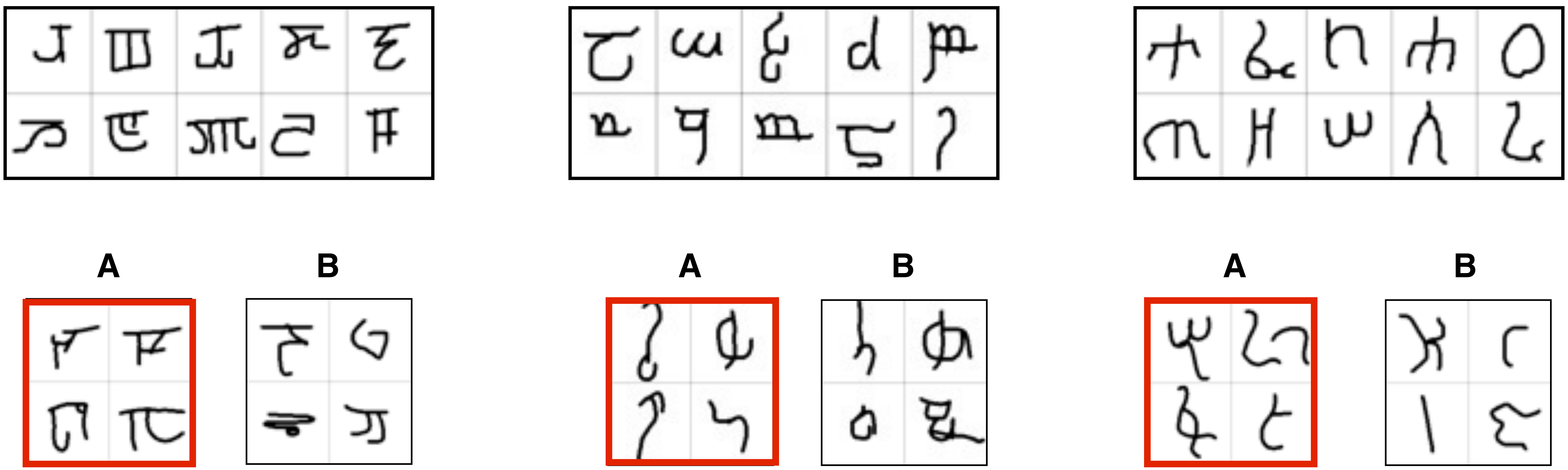
A



B

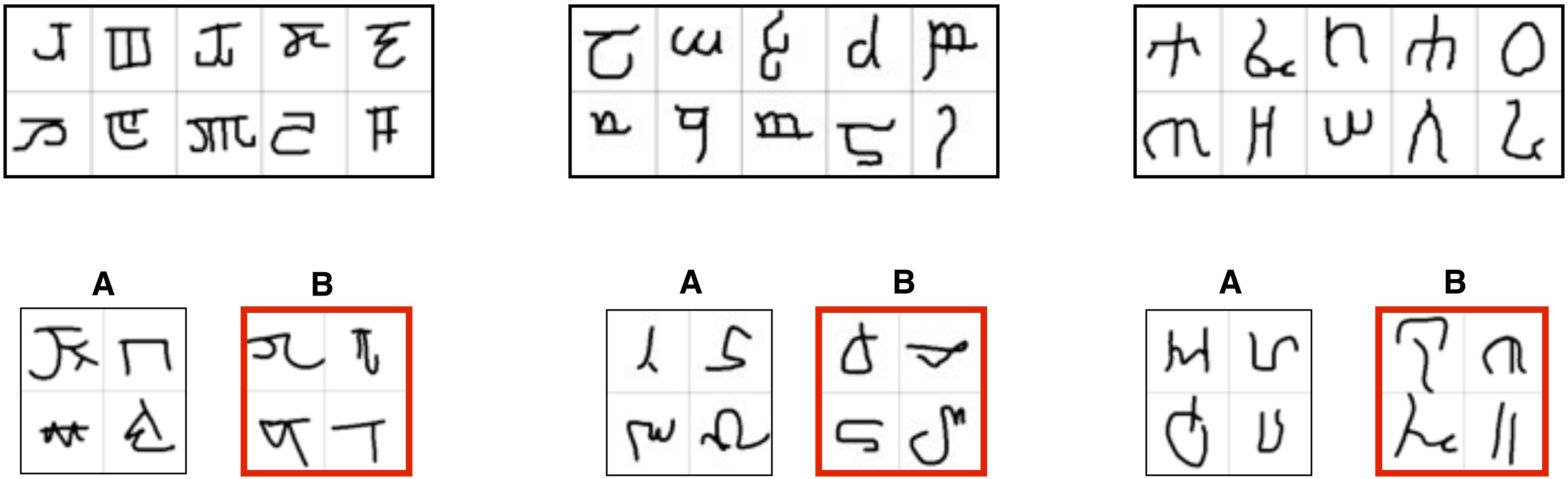
Task: “Design a new character from the same alphabet”

Which grid is produced by the model?



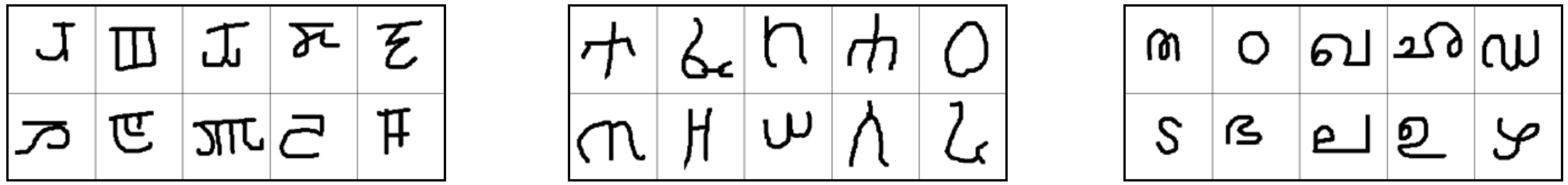
Task: “Design a new character from the same alphabet”

Which grid is produced by the model?

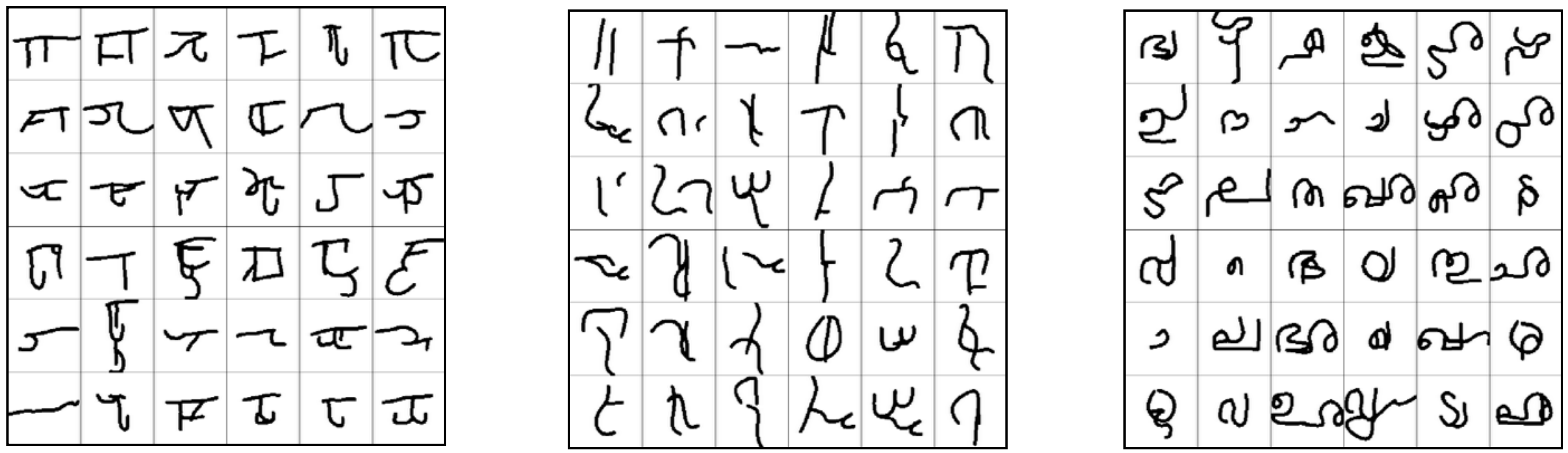


Generate a new characters from the same alphabet

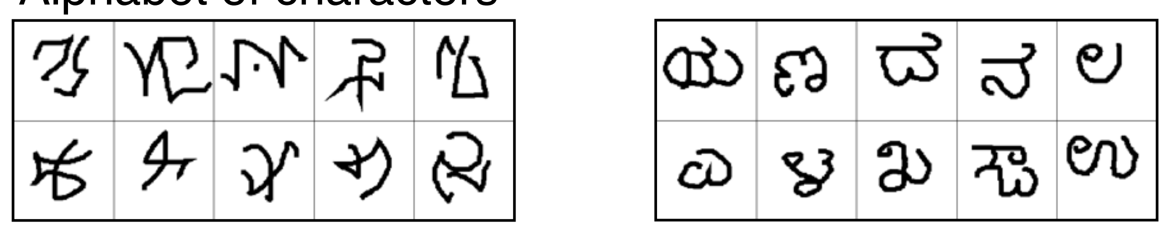
Alphabet of characters



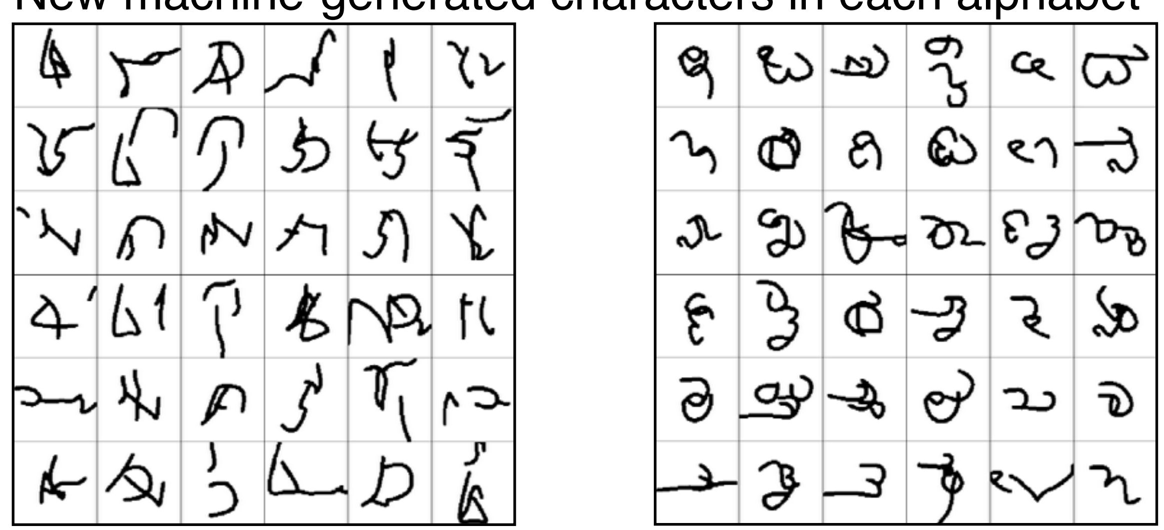
New machine-generated characters in each alphabet



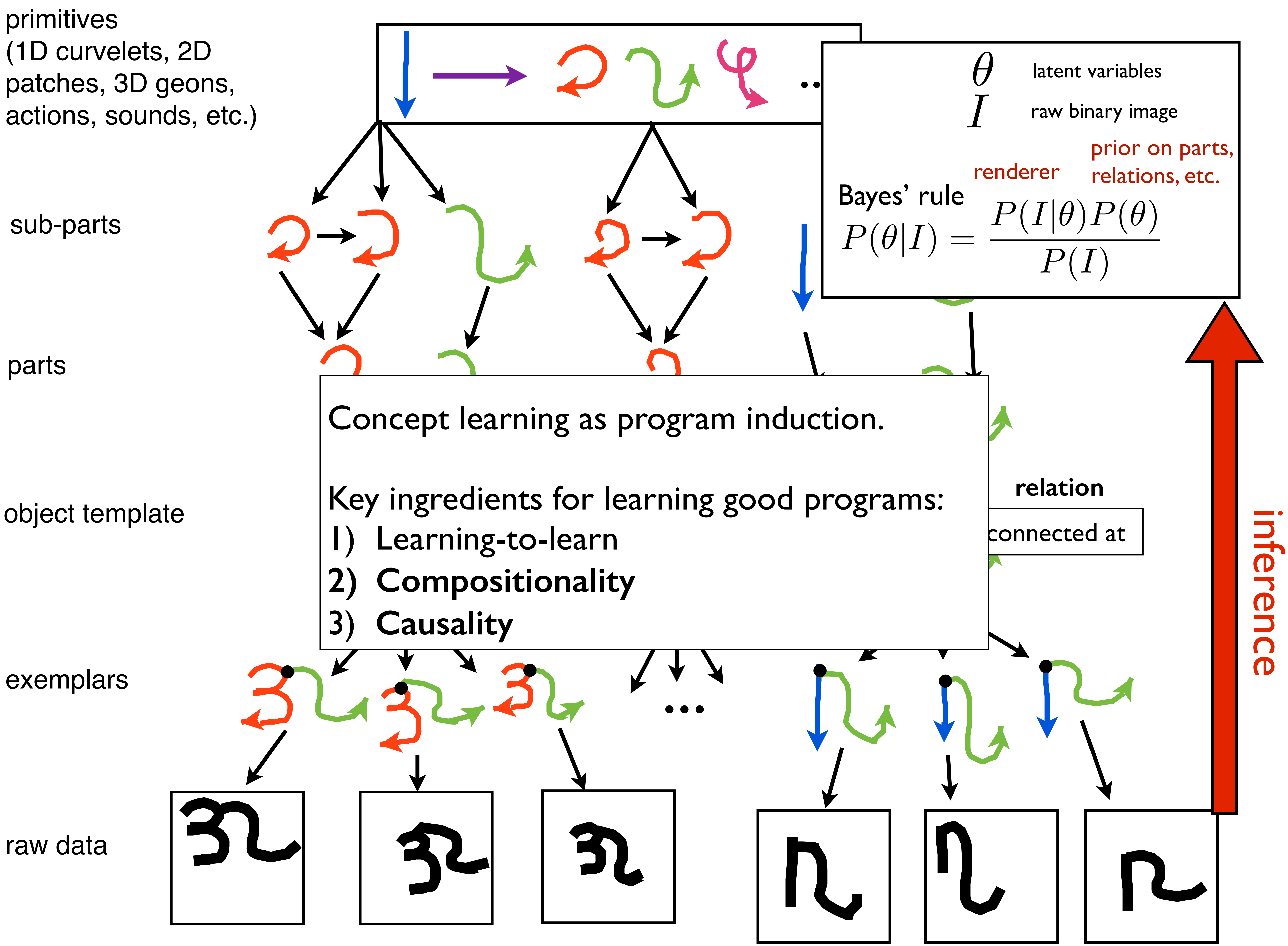
Alphabet of characters



New machine-generated characters in each alphabet



Bayesian Program Learning



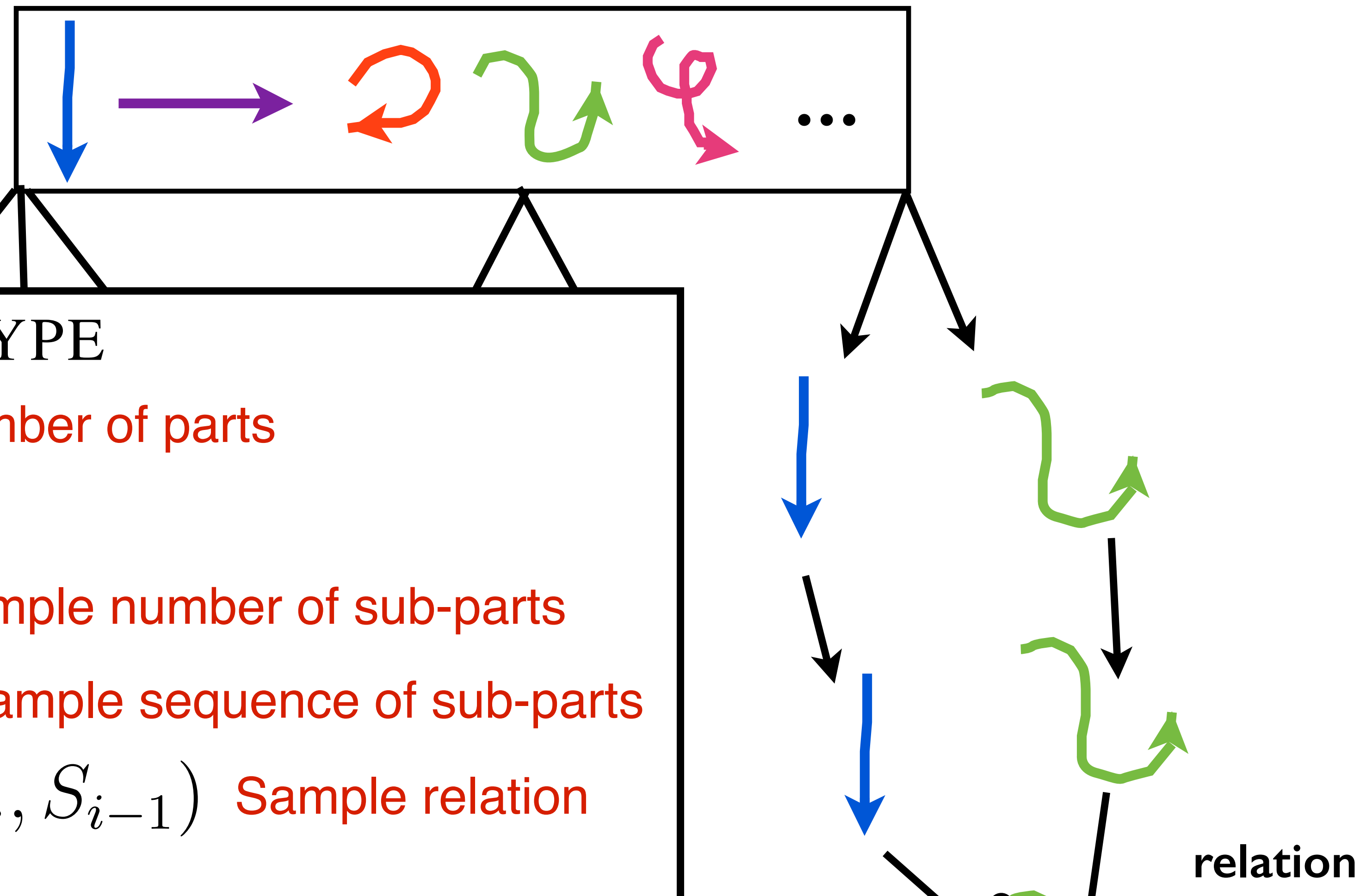
Concept learning as program induction.

Key ingredients for learning good programs:

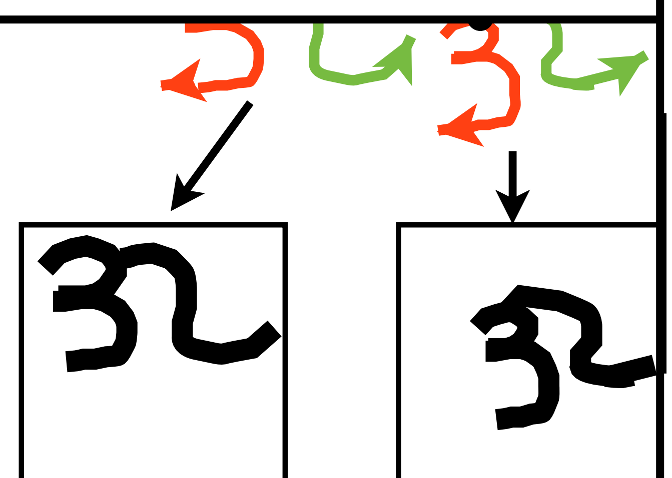
1) Learning-to-learn
2) **Compositionality**
3) **Causality**

Bayesian Program Learning

primitives
(1D curvelets, 2D patches, 3D geons, actions, sounds, etc.)



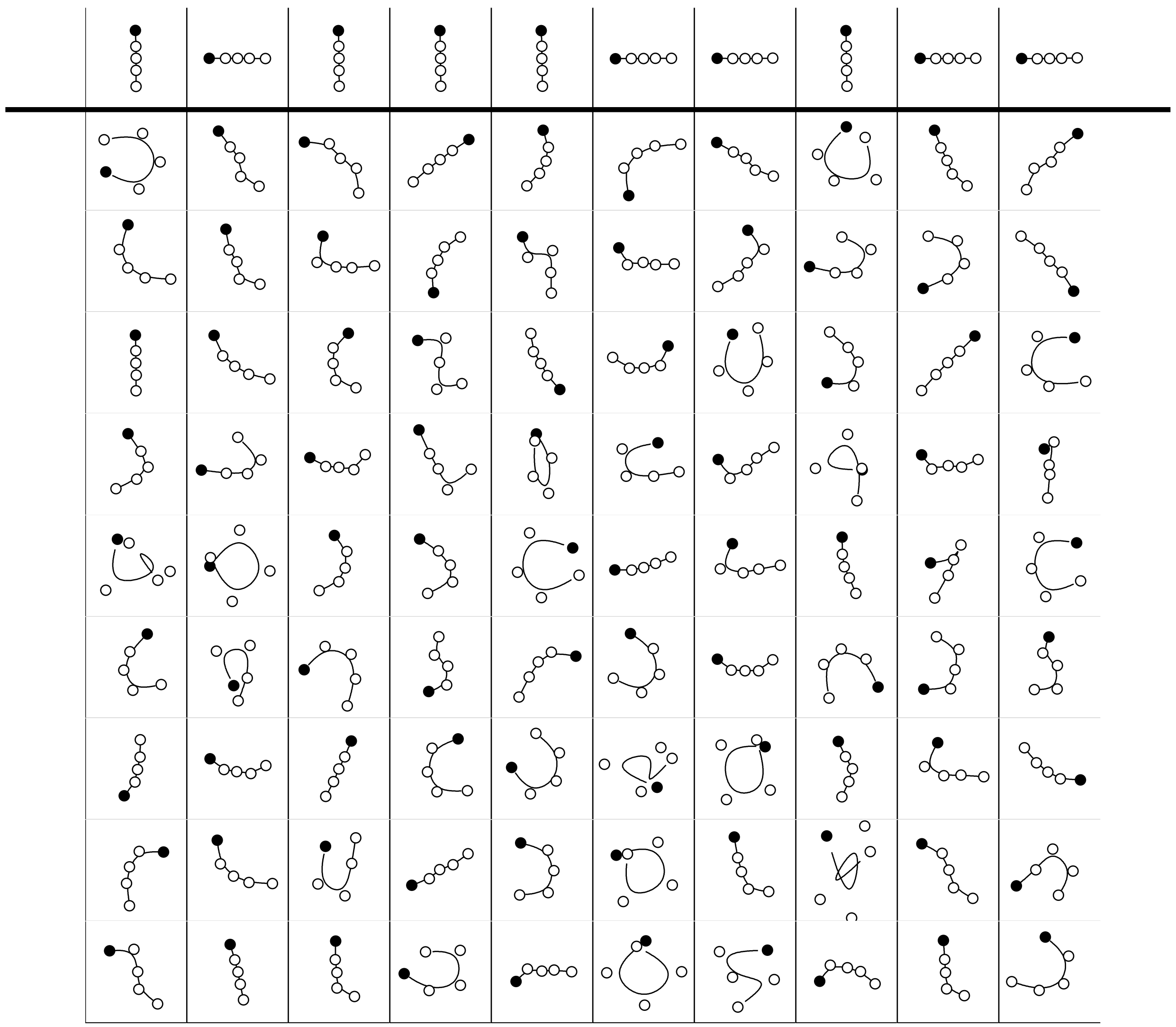
procedure GENERATETYPE
  $\kappa \leftarrow P(\kappa)$ Sample number of parts
  for i = 1 ... κ **do**
    $n_i \leftarrow P(n_i|\kappa)$ Sample number of sub-parts
    $S_i \leftarrow P(S_i|n_i)$ Sample sequence of sub-parts
    $R_i \leftarrow P(R_i|S_1, ..., S_{i-1})$ Sample relation
  end for
  $\psi \leftarrow \{\kappa, R, S\}$
  return @GENERATETOKEN
end procedure   Return handle to a s



procedure GENERATETOKEN(ψ)
  for $i = 1...\kappa$ **do**
    $S_i^{(m)} \leftarrow P(S_i^{(m)}|S_i)$ Add motor variance
    $L_i^{(m)} \leftarrow P(L_i^{(m)}|R_i, T_1^{(m)}, ..., T_{i-1}^{(m)})$ Sample part's start location
    $T_i^{(m)} \leftarrow f(L_i^{(m)}, S_i^{(m)})$ Compose a part's pen trajectory
  end for
  $A^{(m)} \leftarrow P(A^{(m)})$ Sample affine transform
  $I^{(m)} \leftarrow P(I^{(m)}|T^{(m)}, A^{(m)})$
  return $I^{(m)}$ Render and sample the binary image
end procedure

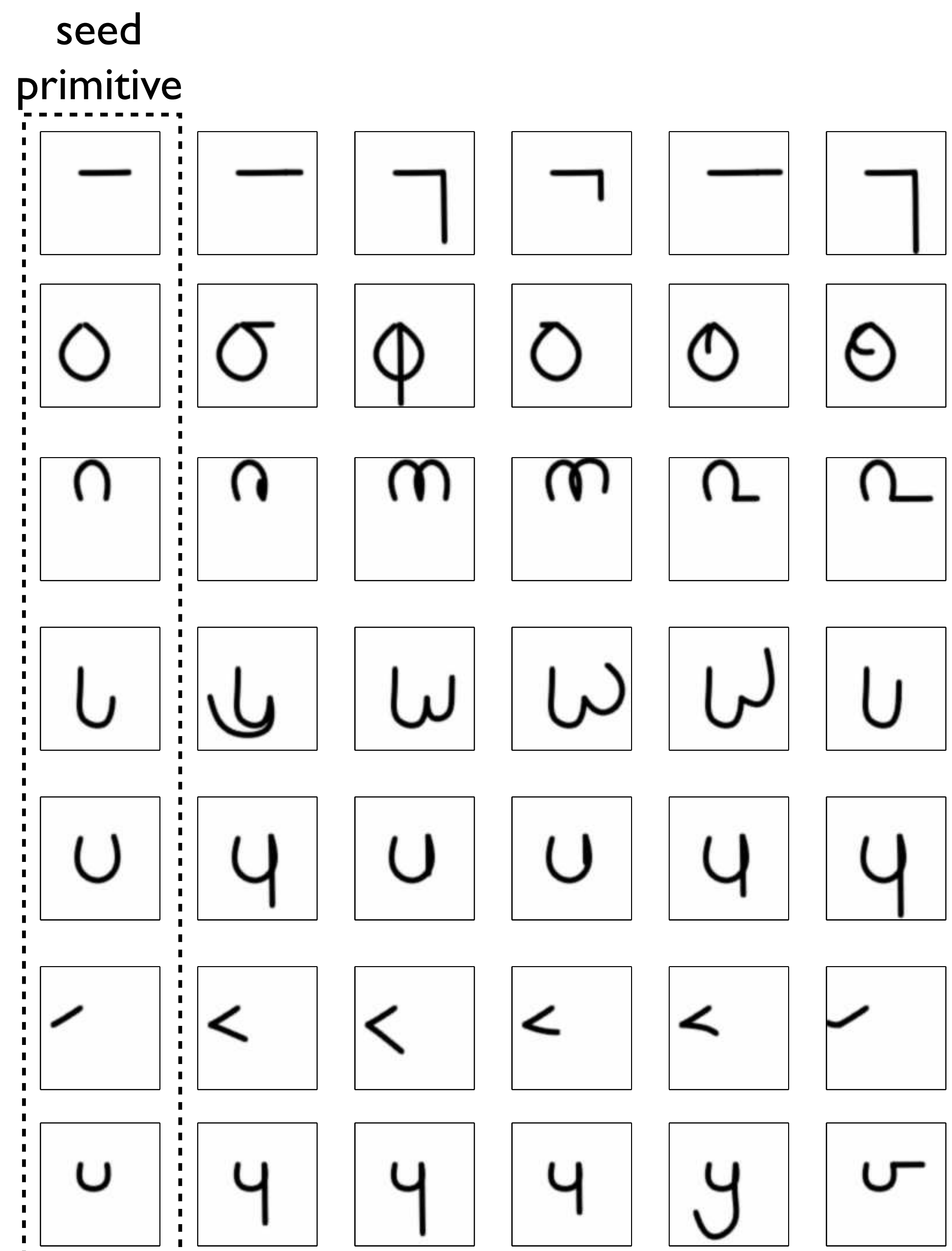
Learning-to-learn programs

learned action primitives



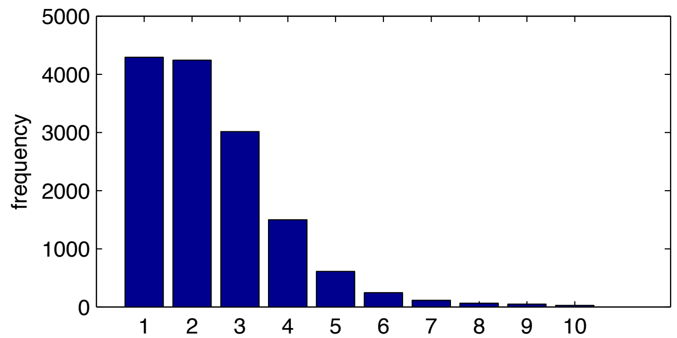
1250 primitives
scale selective
translation invariant

learned primitive transitions

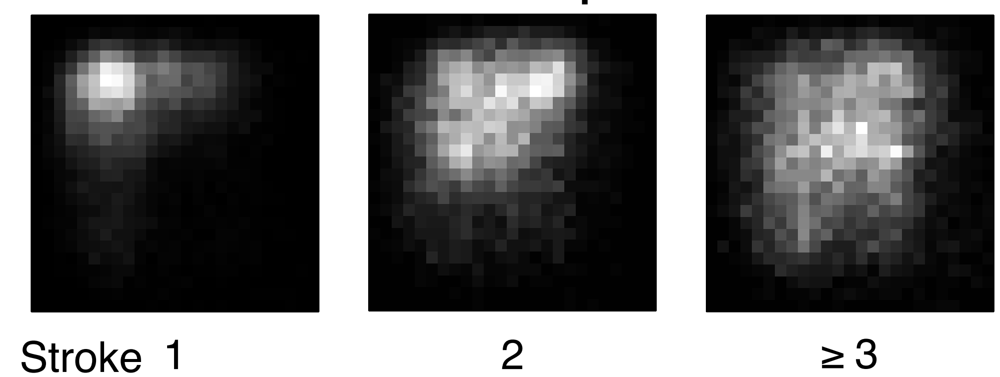


Learning-to-learn programs

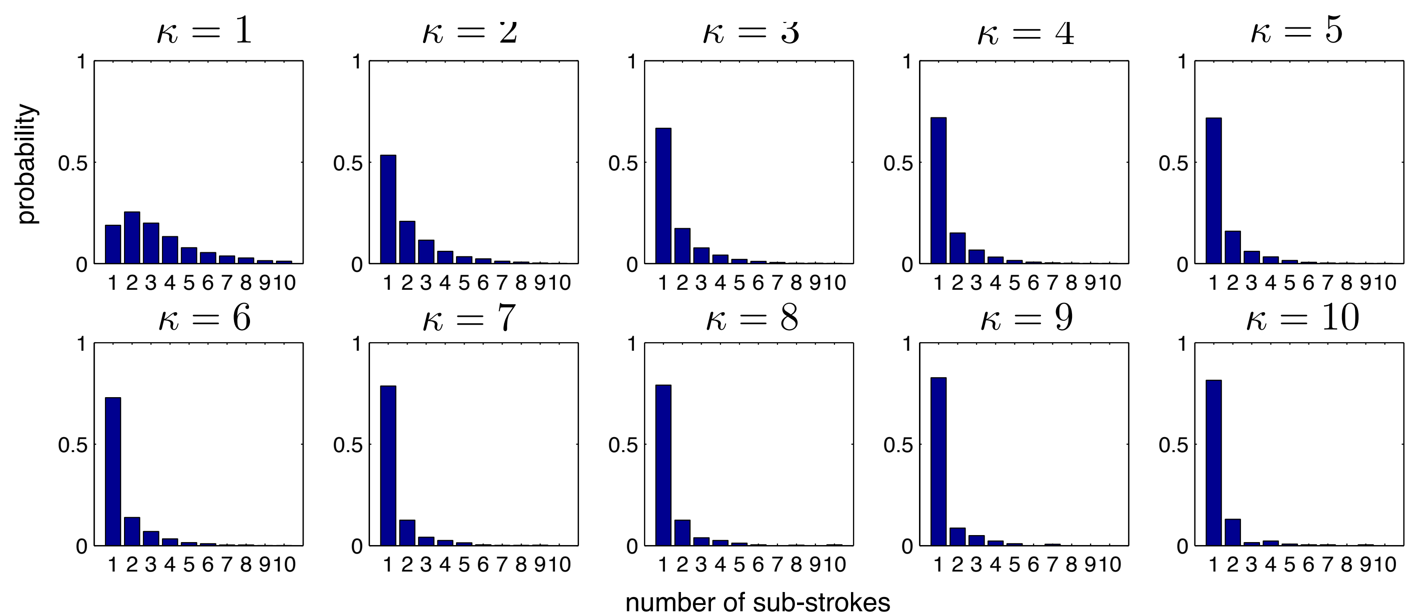
number of strokes



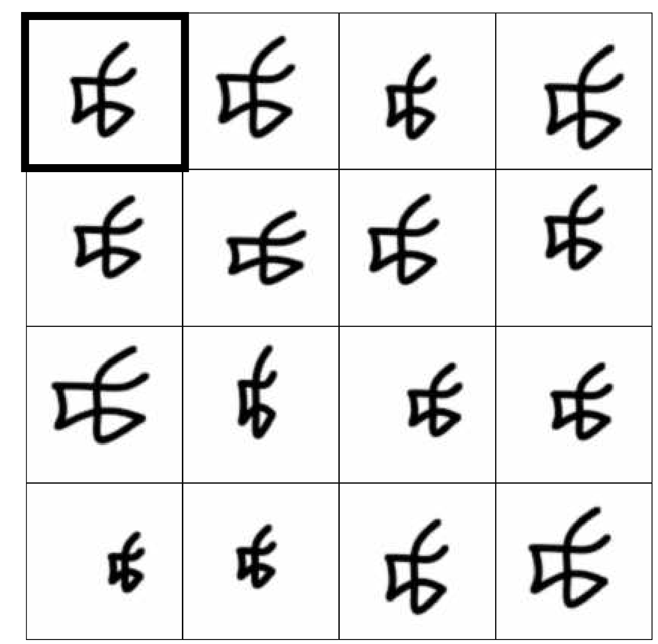
stroke start positions



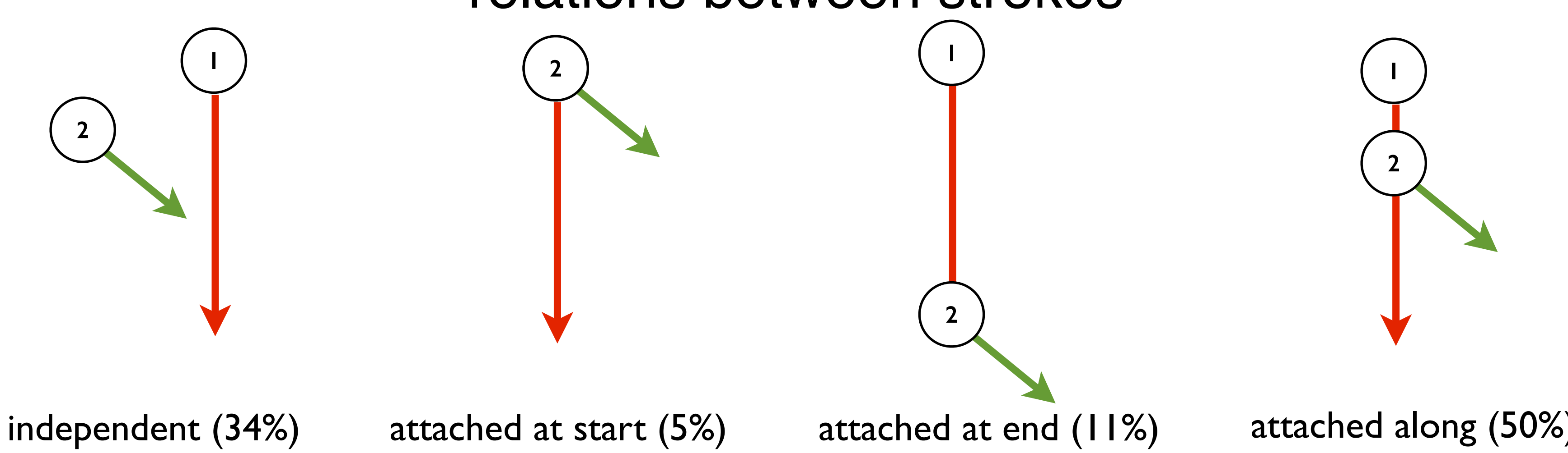
number of sub-strokes for a character with κ strokes



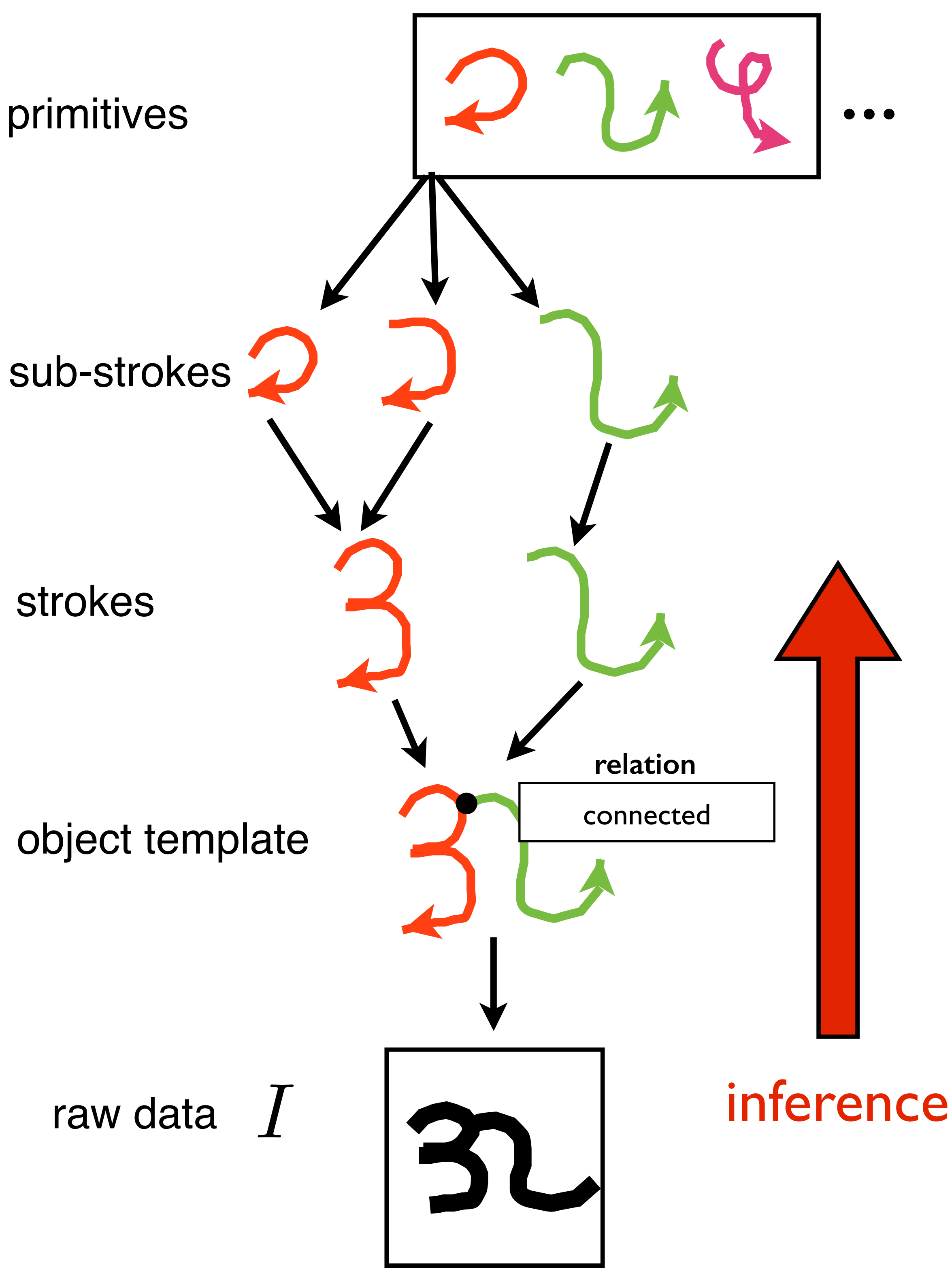
global transformations



relations between strokes



Inferring latent motor programs



θ latent variables

I raw binary image

Bayes' rule

renderer prior on programs

$$P(\theta|I) = \frac{P(I|\theta)P(\theta)}{P(I)}$$

Discrete (K=5) approximation to posterior

$$P(\theta|I) \approx \frac{\sum_{i=1}^{K} w_i \delta(\theta - \theta^{[i]})}{\sum_{i=1}^{K} w_i}$$

such that

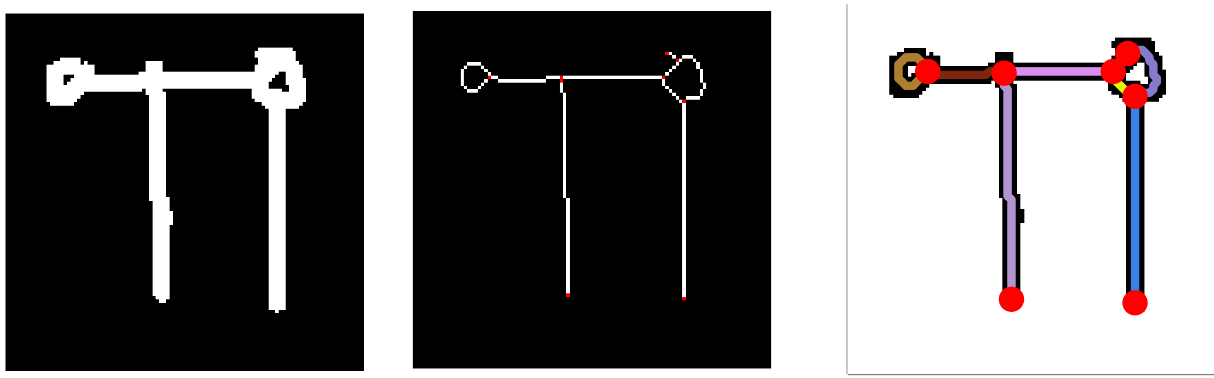
$$w_i \propto P(\theta^{[i]}|I)$$

Intuition: Fit strokes to the observed pixels as closely as possible, with these constraints:

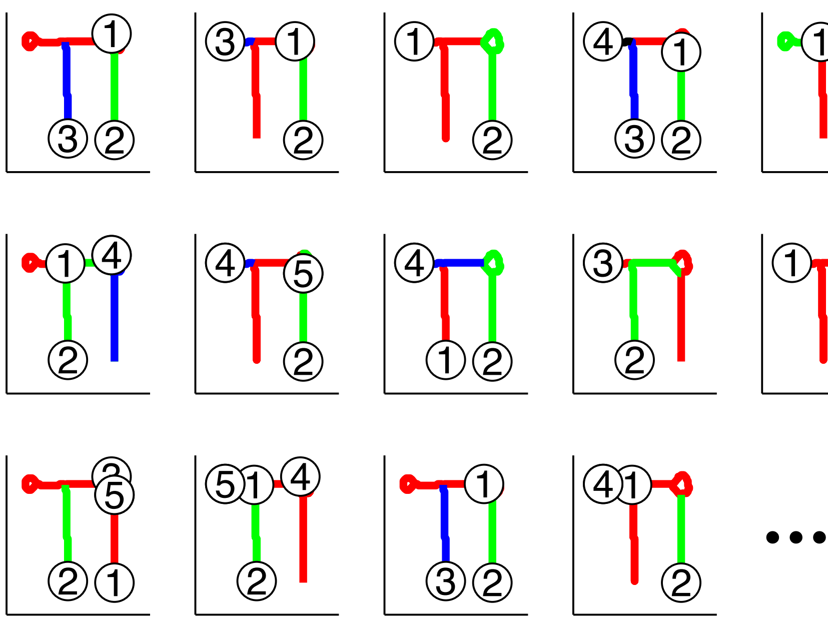
- fewer strokes
- high-probability primitive sequence
- use relations
- stroke order
- stroke directions

Inference

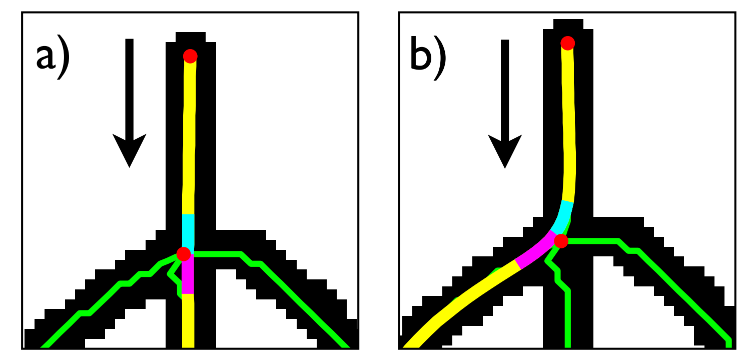
Step 1: characters as undirected graphs



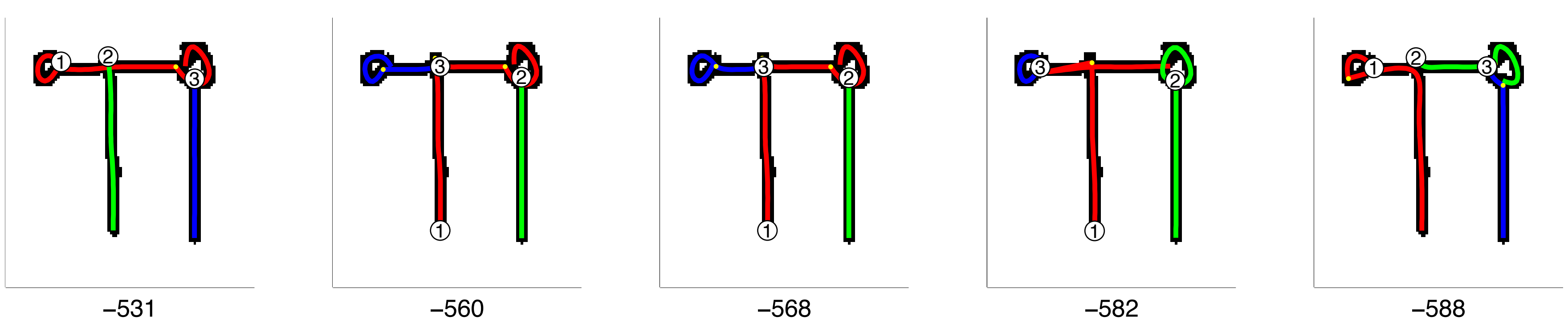
Step 2: guided random parses



more likely less likely



Step 3: Top-down fitting with gradient-based optimization

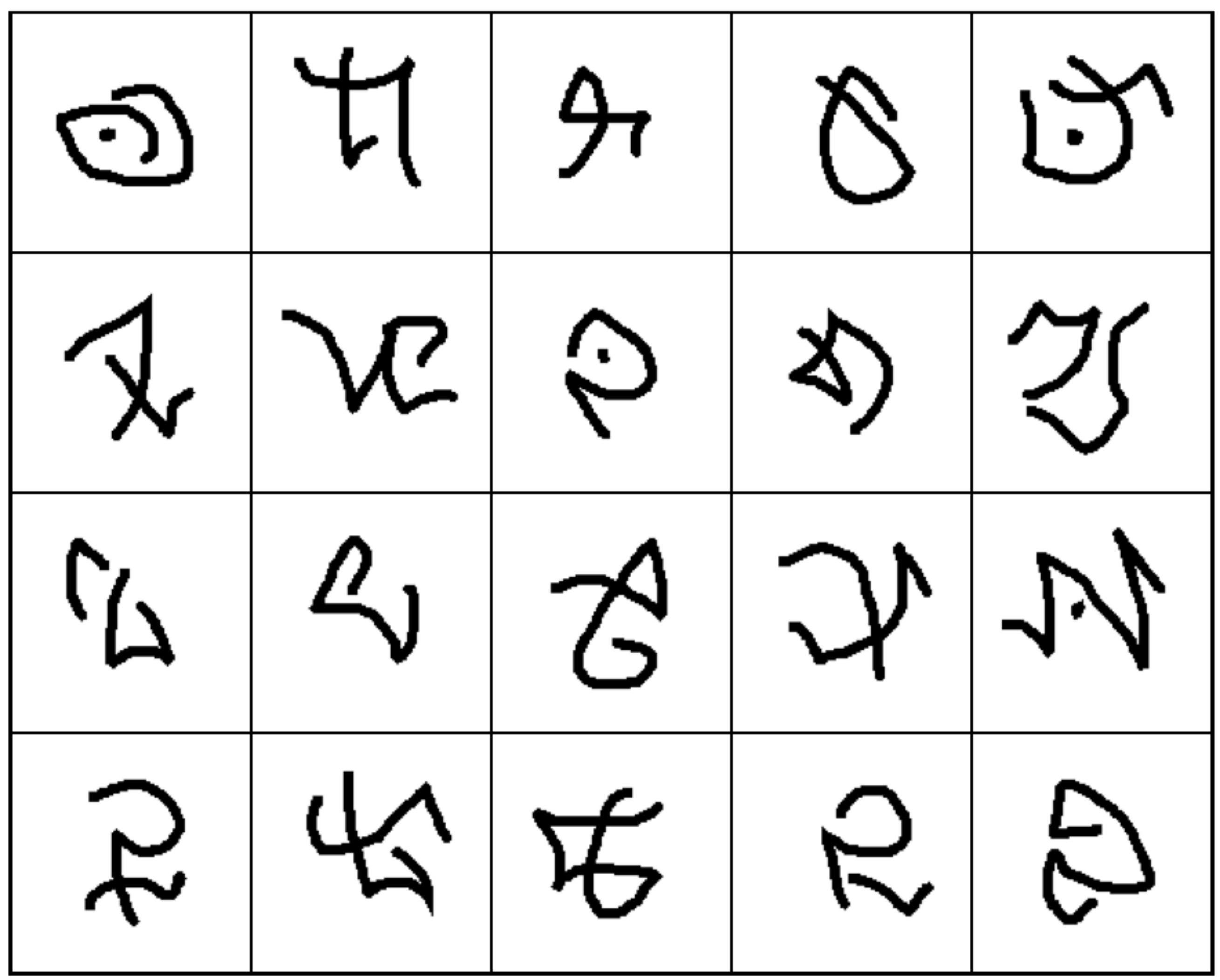


log-probability

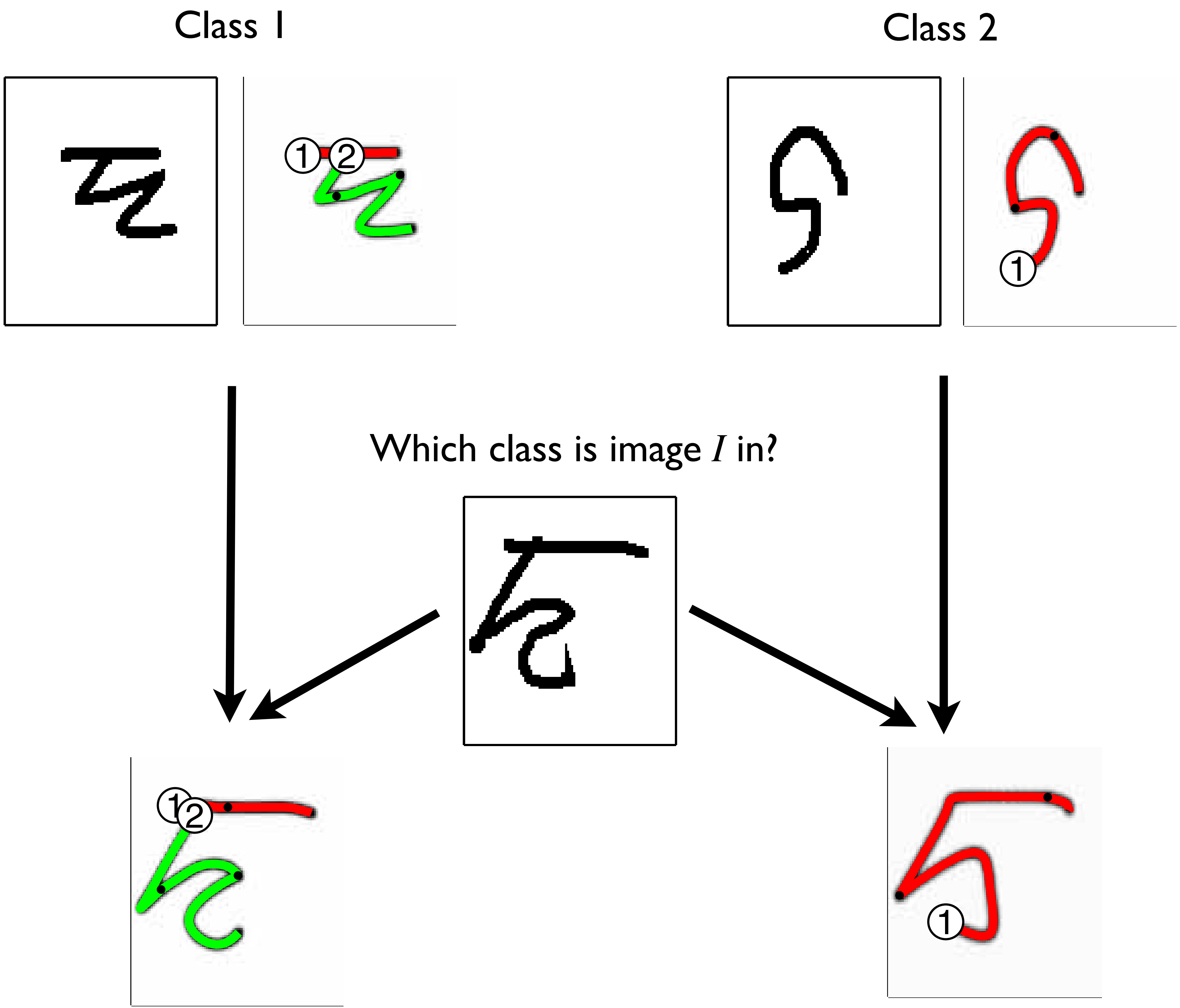
One-shot classification

Target character

Click the image below that shows the same character.



HBPL: Computing the classification score

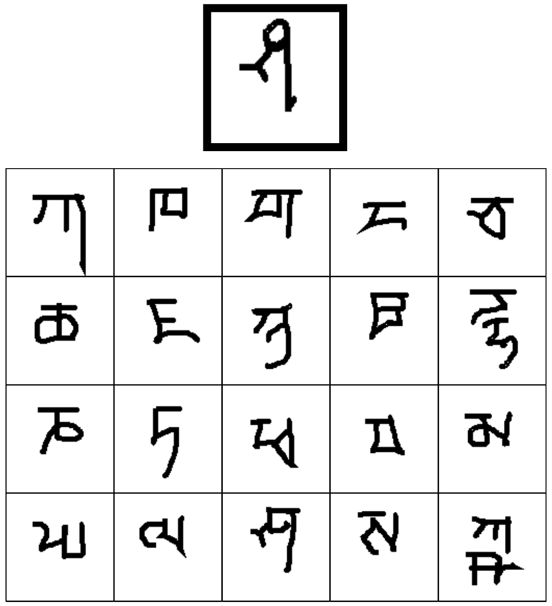


$$\log P(I|\text{class }1) \approx -758$$

$$\log P(I|\text{class }2) \approx -1880$$

Comparing human and machine performance on five tasks

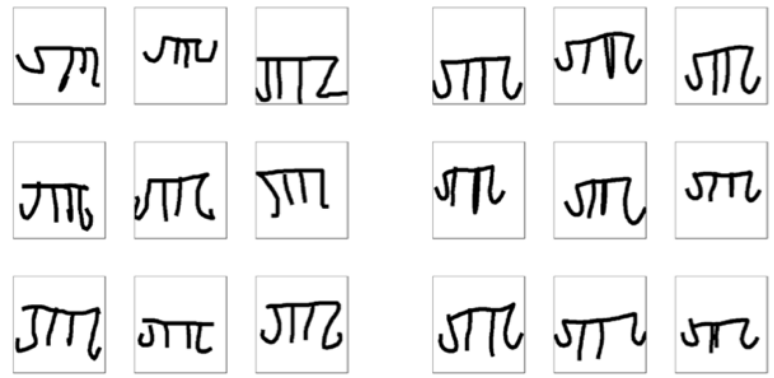
One-shot classification (20-way)



4.5% human error rate
3.2% machine error rate

Generating new examples

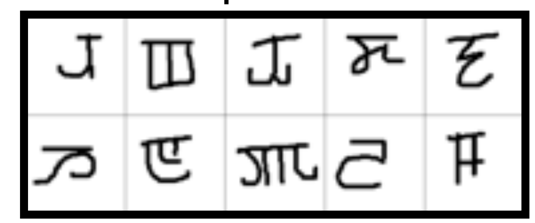
Human or Machine?



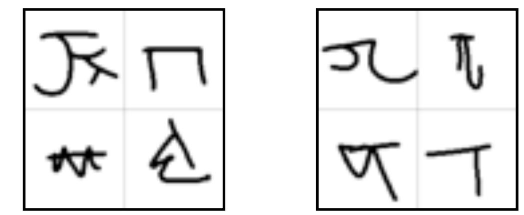
51% Identification (ID) Level
[% judges who correctly ID machine vs. human]

Generating new concepts (from type)

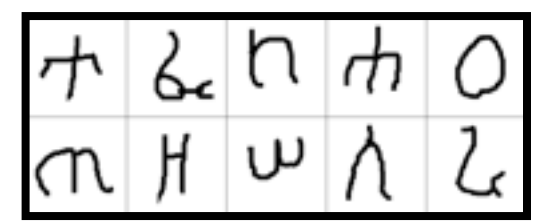
Alphabet



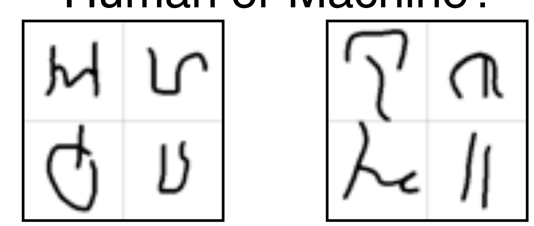
Human or Machine?



Alphabet



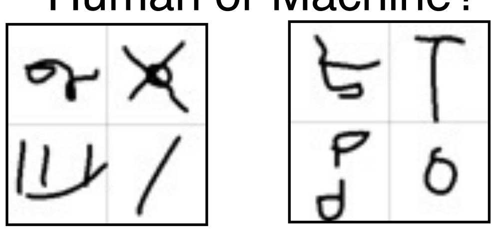
Human or Machine?



49% ID Level

Generating new concepts (unconstrained)

Human or Machine?

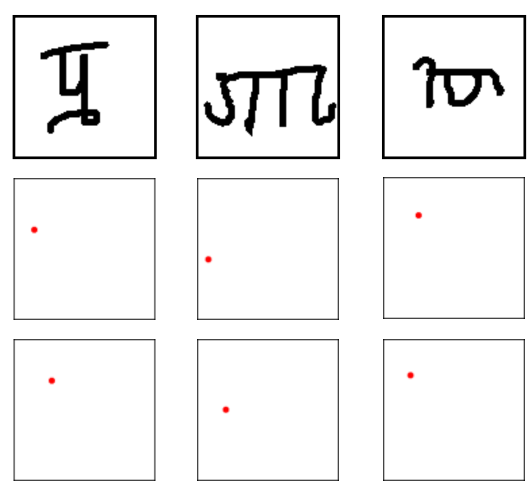


Human or Machine?

51% ID Level

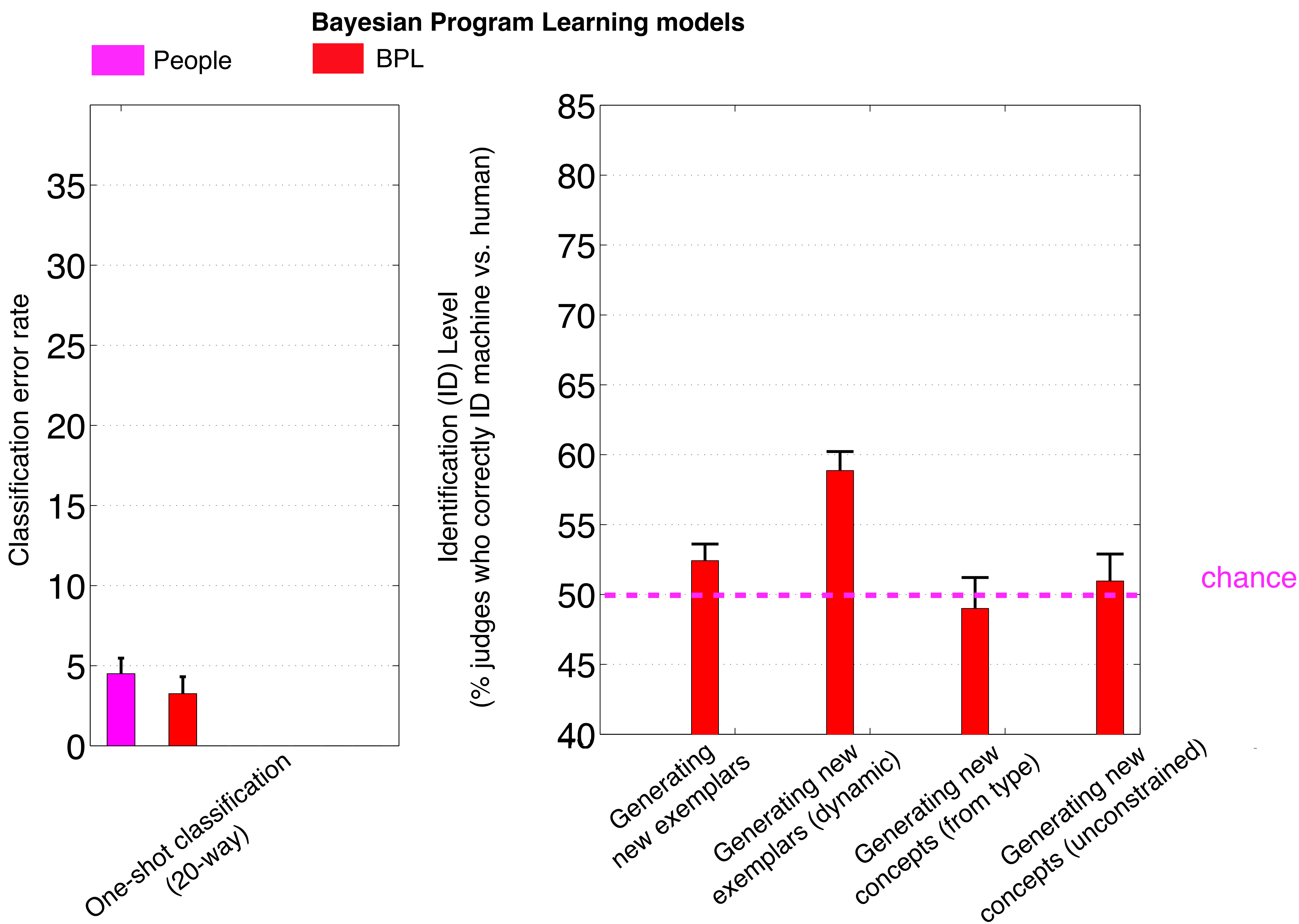
Generating new examples (dynamic)

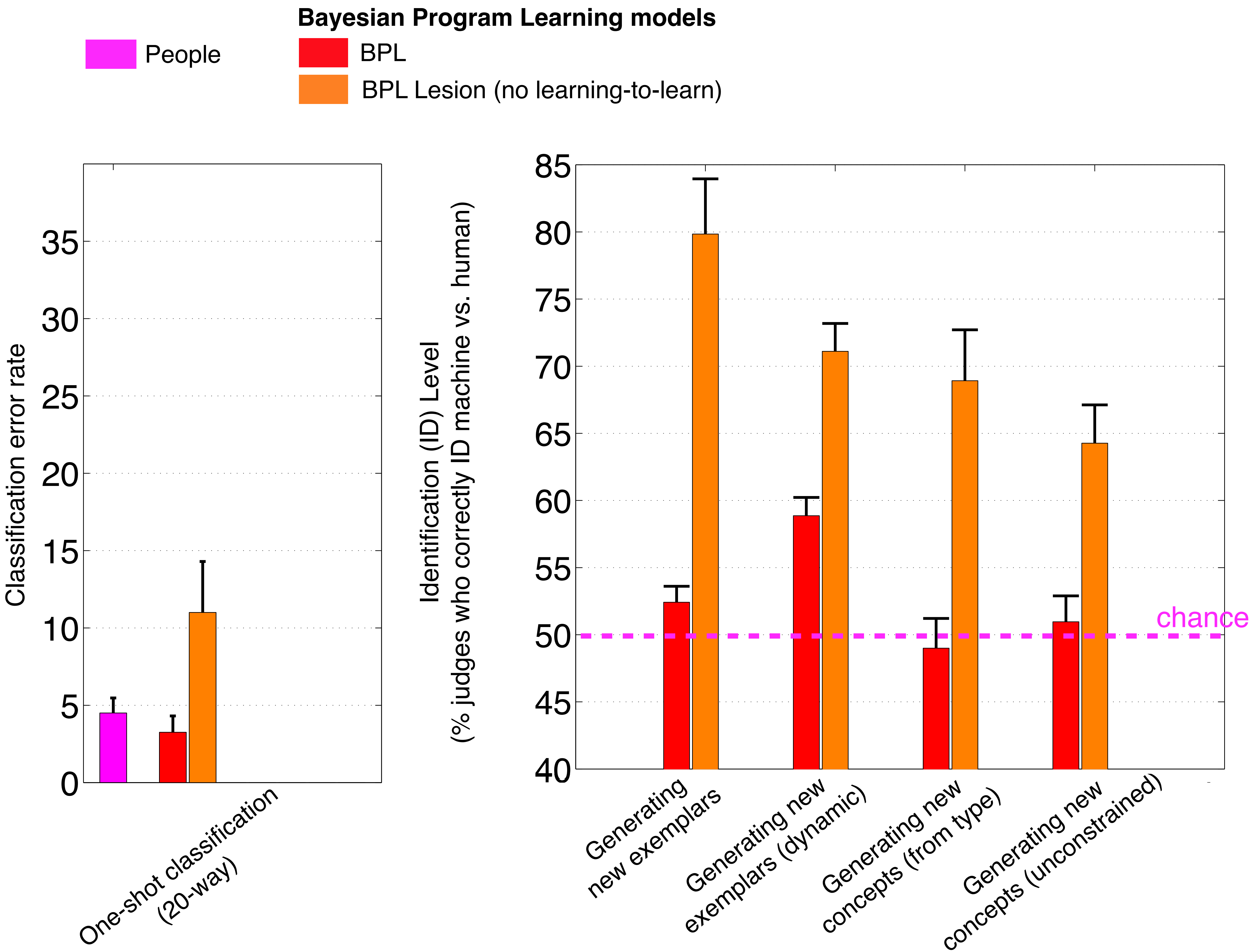
Human or Machine?



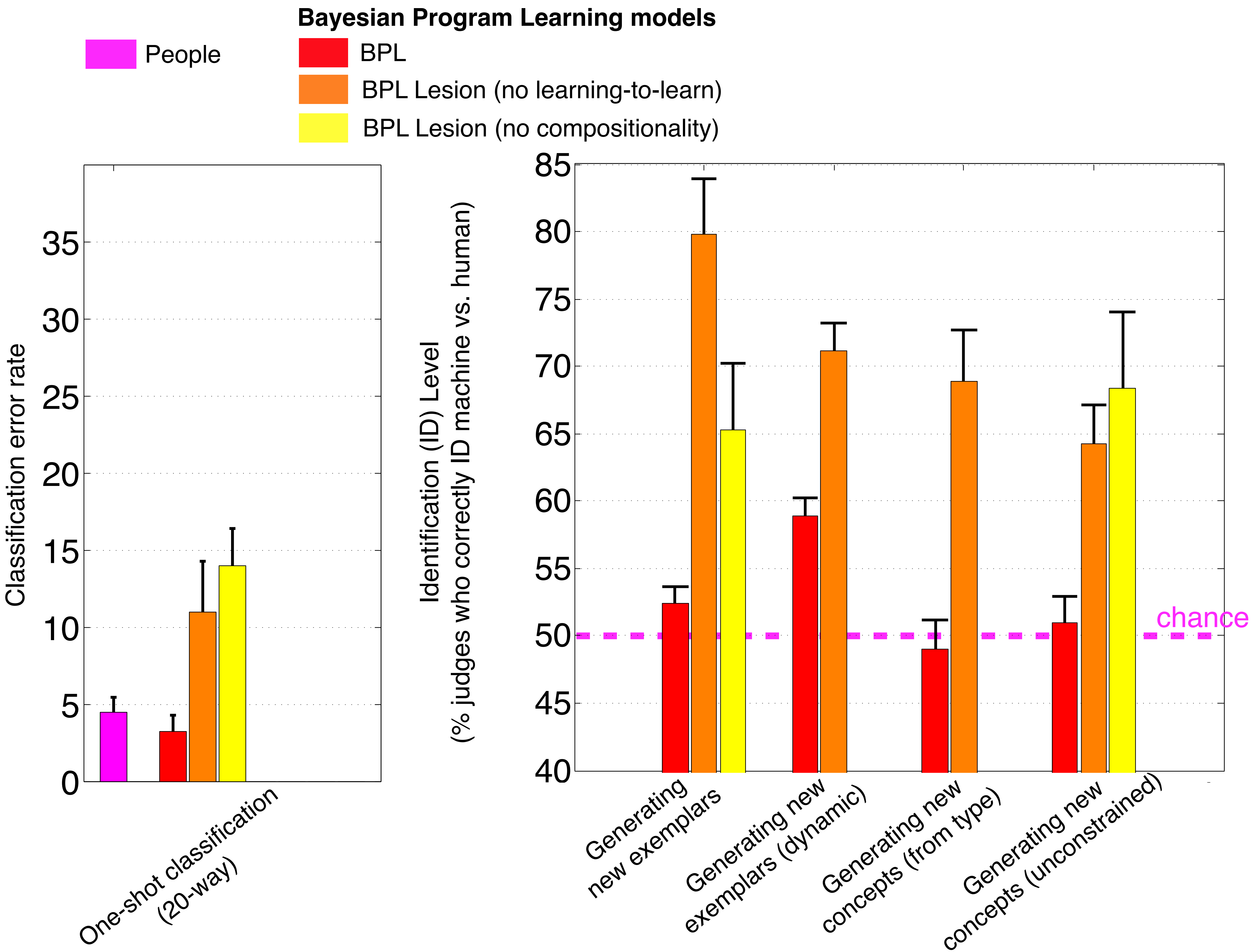
59% ID Level

Analyzing the core ingredients

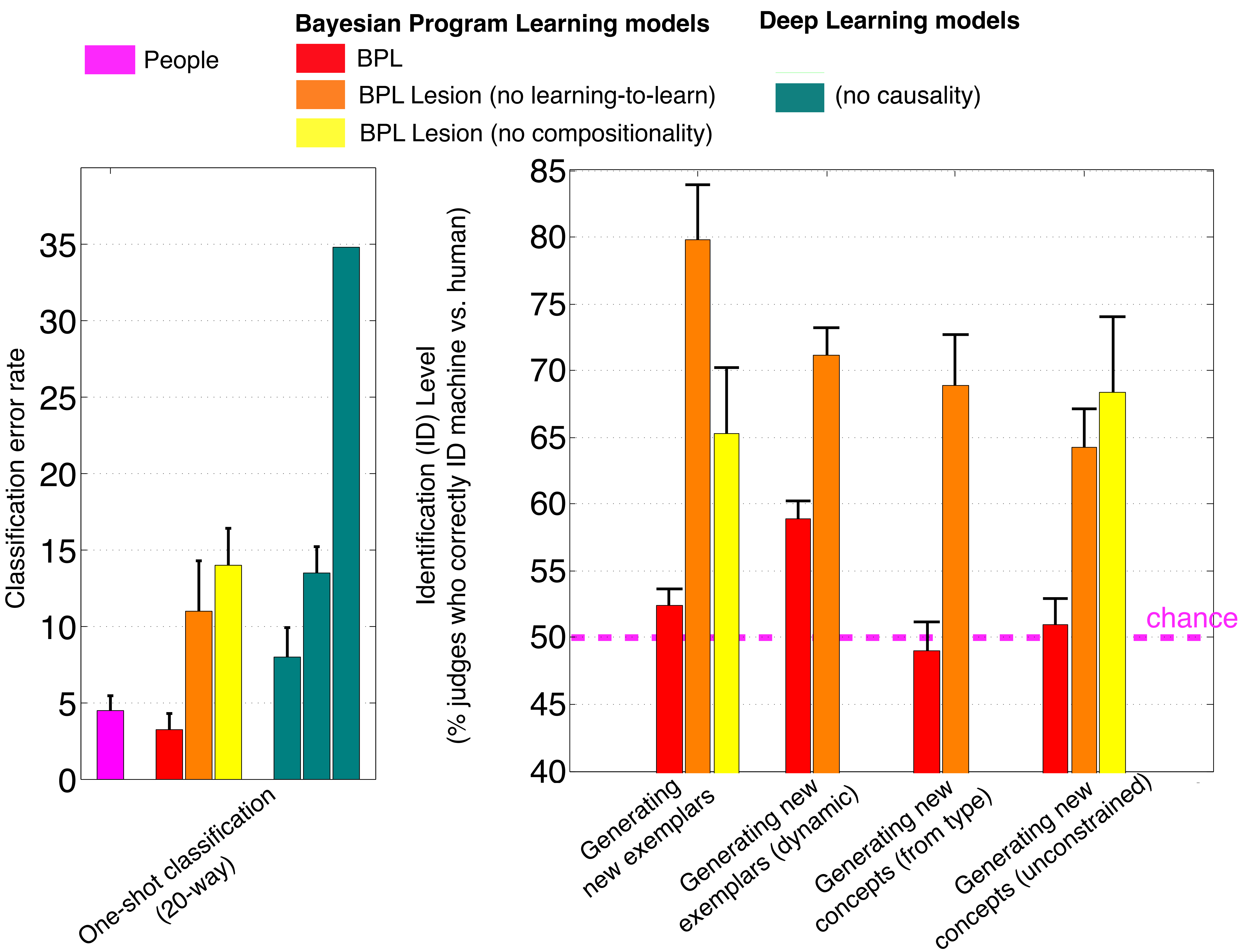




People
Bayesian Program Learning models
BPL
BPL Lesion (no learning-to-learn)
Classification error rate
0
5
10
15
20
25
30
35
One-shot classification (20-way)
Identification (ID) Level
(% judges who correctly ID machine vs. human)
40
45
50
55
60
65
70
75
80
85
chance
Generating new exemplars
Generating new exemplars (dynamic)
Generating new concepts (from type)
Generating new concepts (unconstrained)



Bayesian Program Learning models
People
BPL
BPL Lesion (no learning-to-learn)
BPL Lesion (no compositionality)
Classification error rate
0
5
10
15
20
25
30
35
One-shot classification (20-way)
Identification (ID) Level (% judges who correctly ID machine vs. human)
40
45
50
55
60
65
70
75
80
85
chance
Generating new exemplars
Generating new exemplars (dynamic)
Generating new concepts (from type)
Generating new concepts (unconstrained)

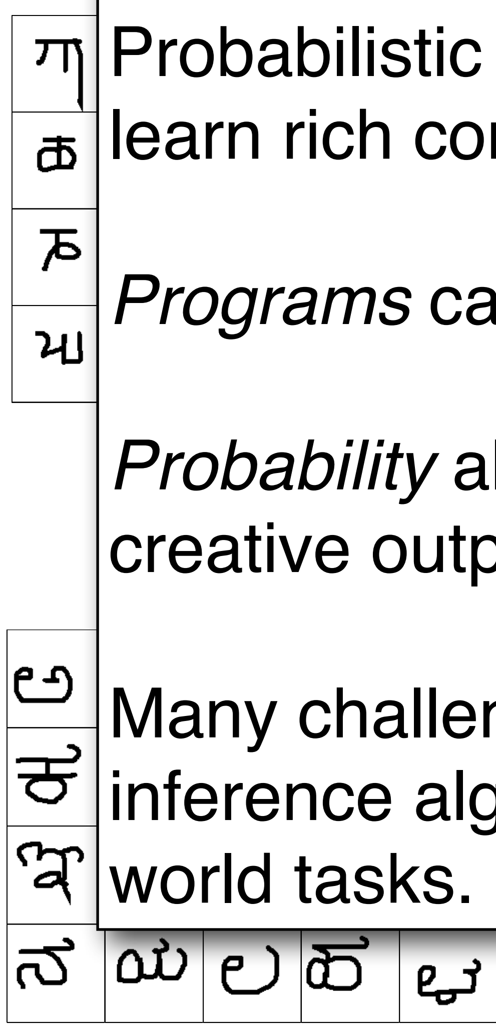


Bayesian Program Learning models
Deep Learning models
People
BPL
BPL Lesion (no learning-to-learn)
BPL Lesion (no compositionality)
(no causality)
Classification error rate
One-shot classification (20-way)
Identification (ID) Level (% judges who correctly ID machine vs. human)
chance
Generating new exemplars
Generating new exemplars (dynamic)
Generating new concepts (from type)
Generating new concepts (unconstrained)

How can people acquire such *rich concepts* from only *one or a few examples*?

the speed of learning

the richness of representation



Conclusion

Probabilistic programs can help us understand how people learn rich concepts from sparse data.

Programs can represent abstract causal processes.

Probability allows models to handle noise and produce creative outputs.

Many challenges for future work, including developing new inference algorithms and extending approach to other real world tasks.