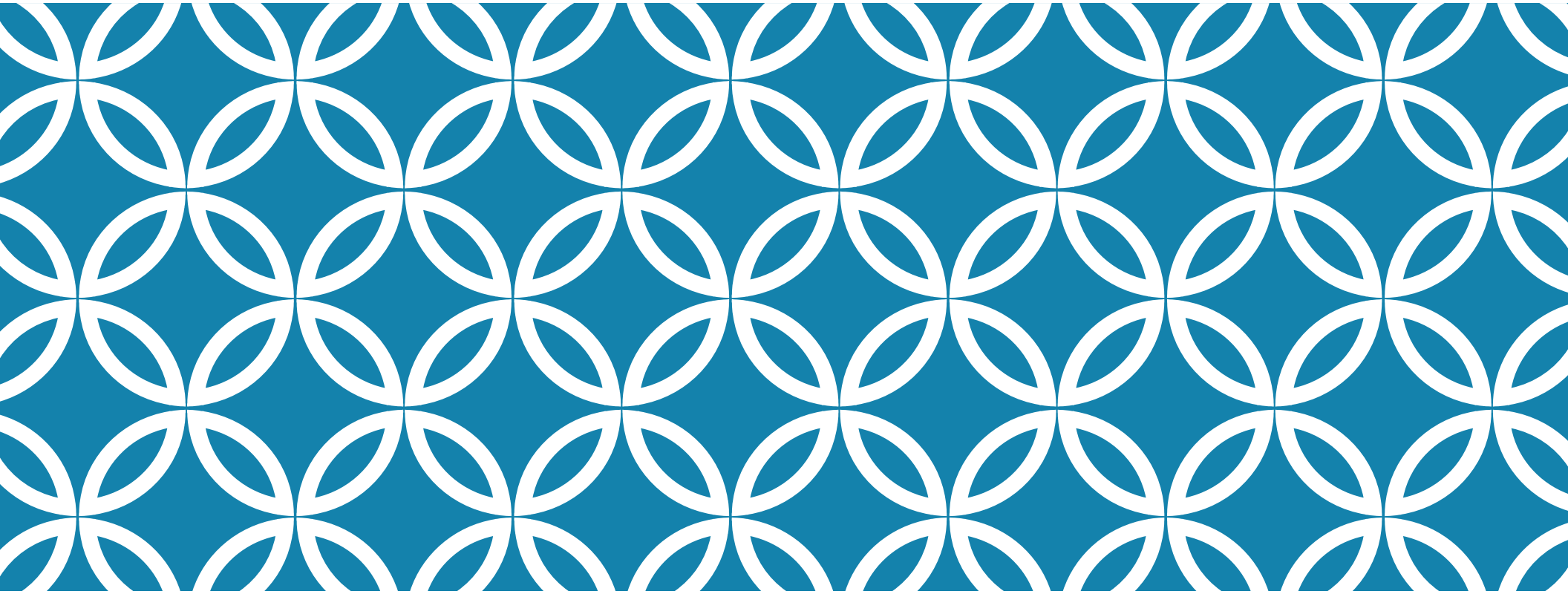


LEARNING TO INFER GRAPHICS PROGRAMS FROM HAND DRAWN IMAGES

Kevin Ellis - MIT , Daniel Ritchie - Brown University, Armando Solar-Lezama - MIT , Joshua b. Tenenbaum - MIT

Presented by : Maliha Arif

Advanced Computer Vision -CAP 6412

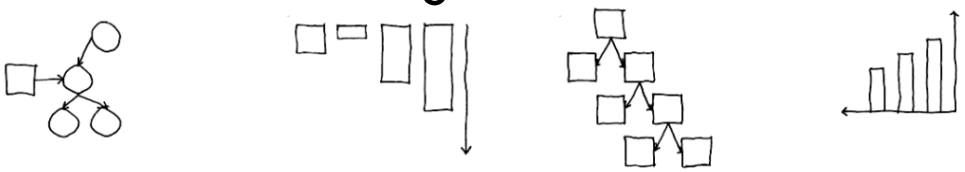
29th March 2018

OUTLINE

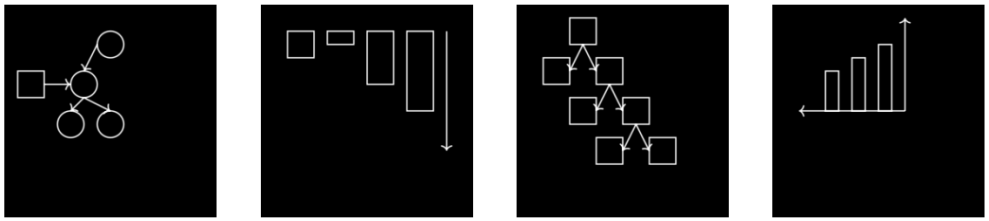
- Introduction / Motivation
- Road map – Trace Hypothesis
- Stage 1 – Obtaining a Trace Set using Neural Network
  - Training
  - Generalization to Hand Drawings
- Stage 2 – Obtaining a Program from Trace set
  - Search Algorithm
  - Compression Factor
- Applications

MOTIVATION

Hand Drawn Images



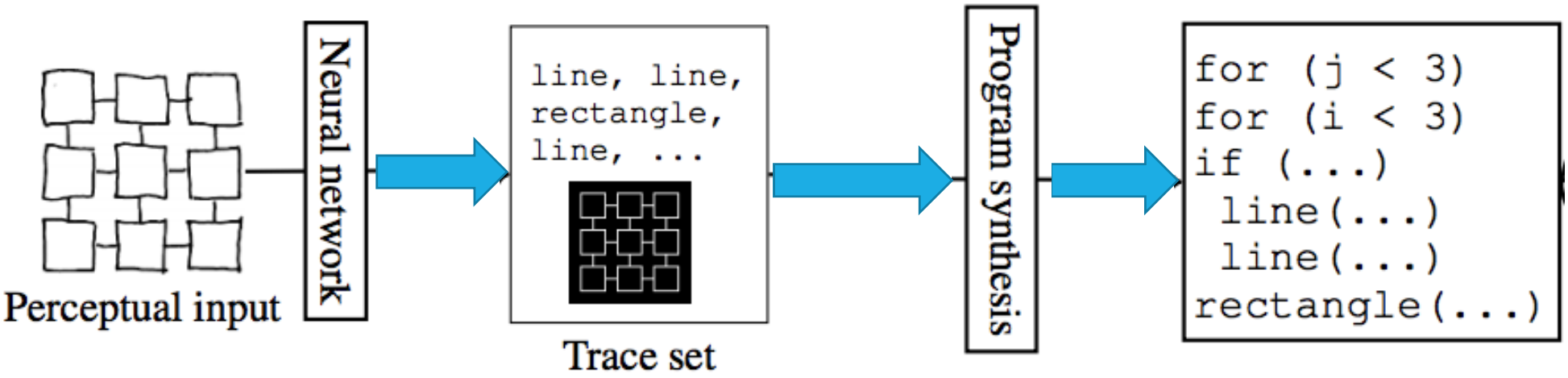
Trace Set- Rendered in Latex



Synthesized Program

```
for (i < 3)
 rectangle(3*i,-2*i+4,
           3*i+2,6)
 for (j < i + 1)
  circle(3*i+1,-2*j+5)
```

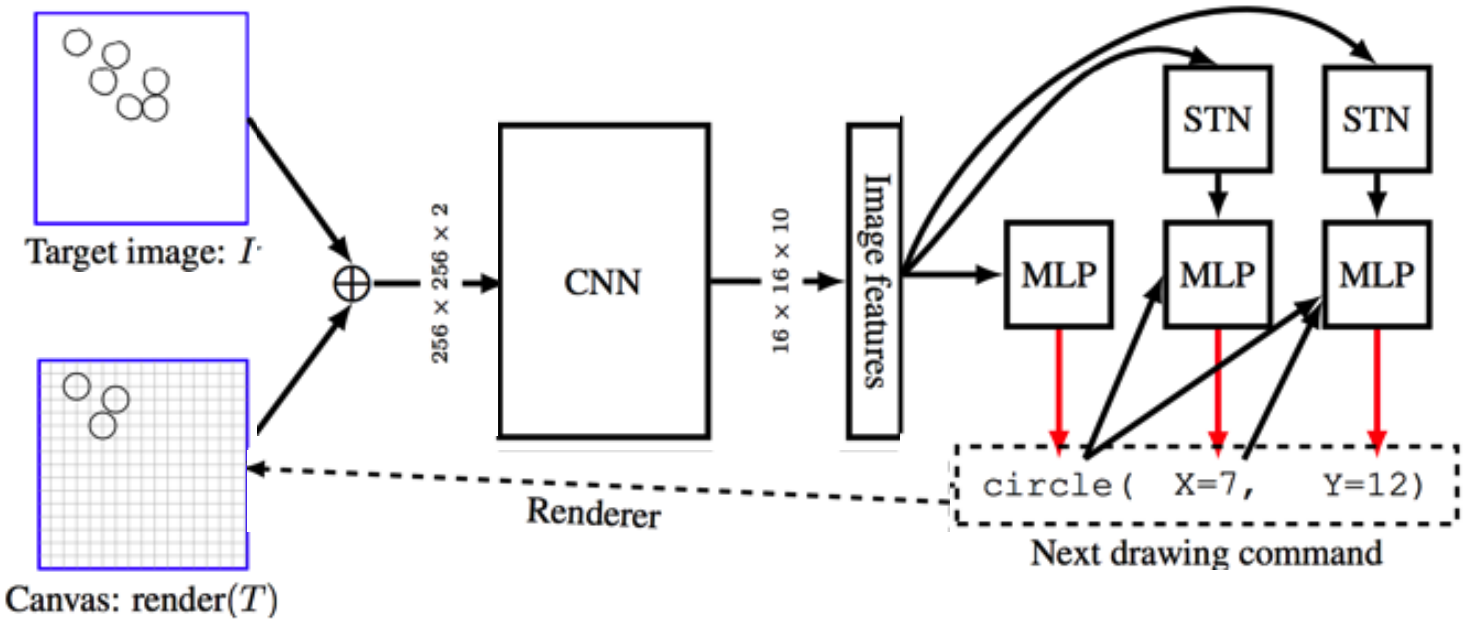
TRACE HYPOTHESIS



Stage 1

Stage 2

STAGE 1: NEURAL NETWORK



NEURAL NETWORK COMPONENTS

- CNN layers
  - Layer 1: 20 8 × 8 convolutions, 2 16 × 4 convolutions, 2 4 × 16 convolutions. Followed by 8 × 8 pooling with a stride size of 4.
  - Layer 2: 10 8 × 8 convolutions. Followed by 4 × 4 pooling with a stride size of 4.
- Drawing commands are predicted Token by Token using MLPs and STNs
- Example Circle command is predicted , then X coordinate then Y coordinate

NEURAL NETWORK

- Circle command or first token is predicted using multinomial logistic regression using a Softmax layer

$$\mathbb{P}[t_1] \propto \exp\left(W_{t_1} f + b_{t_1}\right)$$

- Subsequent Tokens are predicted using a Differentiable Attention Mechanism and previously predicted Tokens
- STN (Spatial Transformer Network)

STN- SPATIAL TRANSFORMER NETWORK

- CNNs lack the ability to be spatially invariant in a computationally and parameter efficient manner.
- Max Pooling layers in CNNs satisfy this property where the receptive fields are fixed and local.
- STN (Spatial Transformer Network) is a dynamic mechanism that can actively spatially transform an image or feature map.

Spatial Transformer Networks : Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu, CVPR 2015

STN – SPATIAL TRANSFORMER NETWORKS

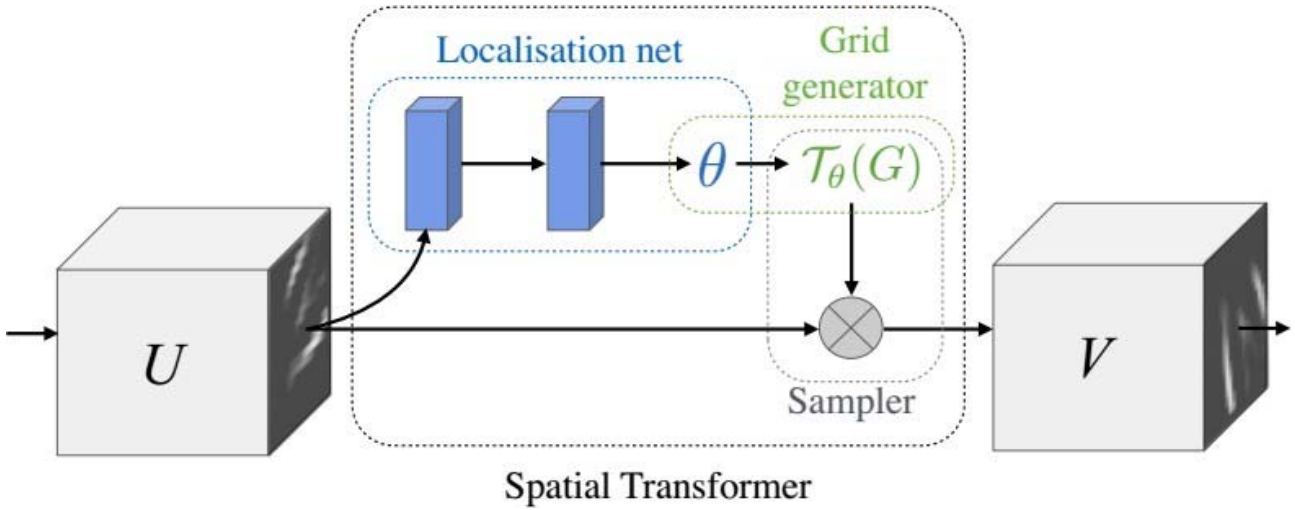
- Spatial transformers can be incorporated into CNNs to benefit multifarious tasks such as:
  - Image Classification
  - Co-localization
  - Spatial attention

We use them here for spatial attention.

Spatial Transformer Networks : Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu, CVPR 2015

STN – SPATIAL TRANSFORMER NETWORKS

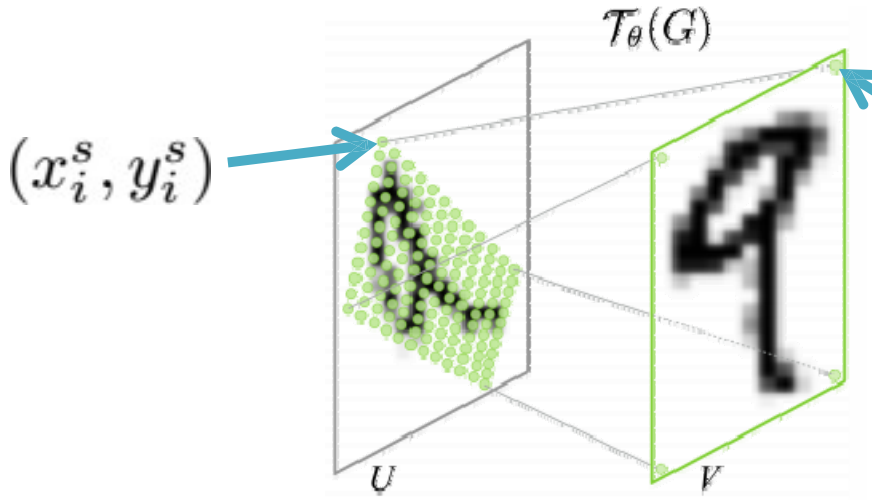
- The Spatial transformer mechanism is split into 3 parts



Spatial Transformer Networks : Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu, CVPR 2015

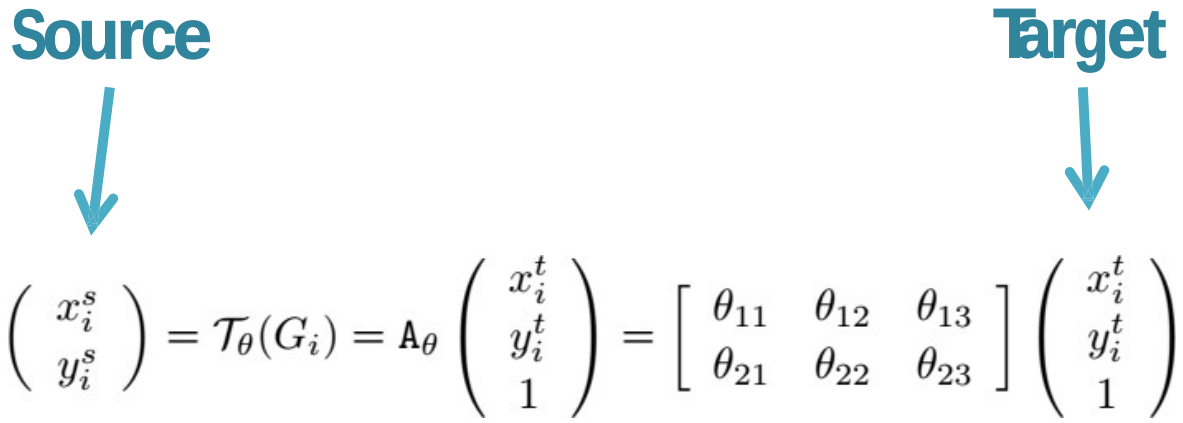
STN – SPATIAL TRANSFORMER NETWORKS

- Grid Generator + Sampler : Affine transform



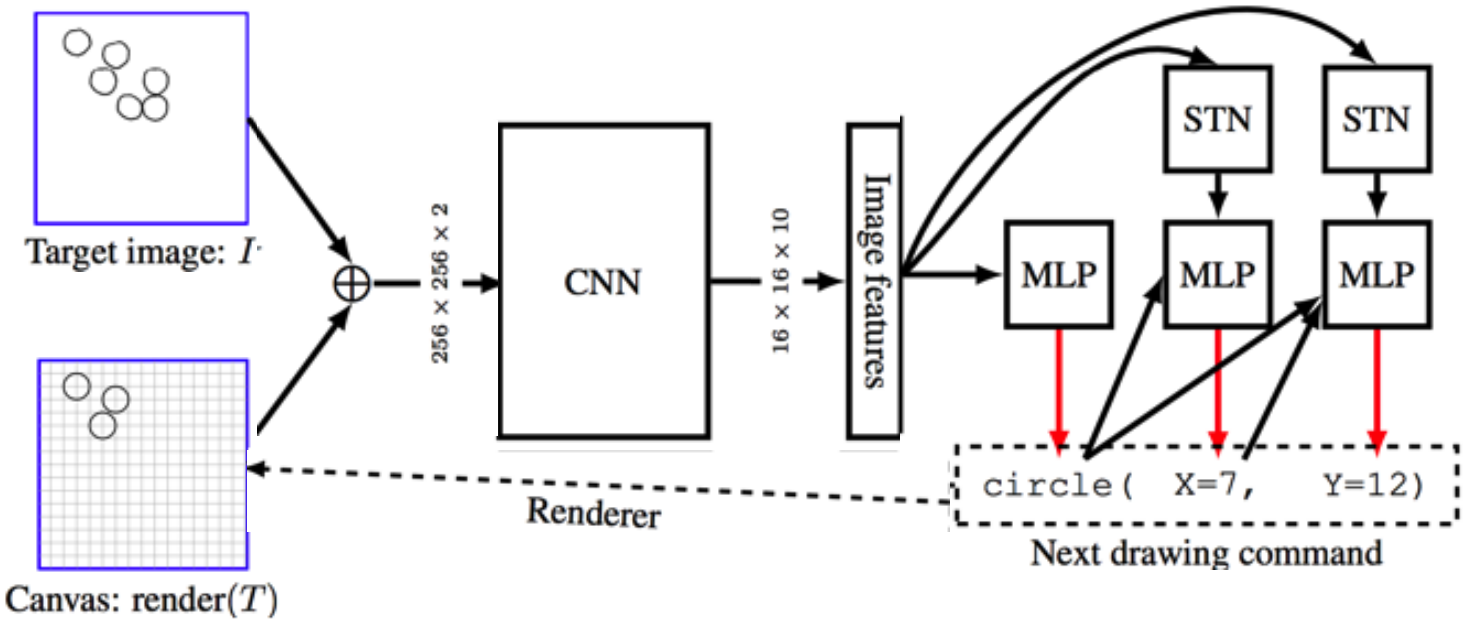
Spatial Transformer Networks : Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu, CVPR 2015

STN – SPATIAL TRANSFORMER NETWORKS



Spatial Transformer Networks : Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu, CVPR 2015

STAGE 1: NEURAL NETWORK



PROBABILITY DISTRIBUTION OF THE CNN

- Distribution over next drawing command

$$\mathbb{P}_\theta[t_1 t_2 \cdots t_K | I, T] = \prod_{k=1}^{K} \mathbb{P}_\theta \left[t_k | a_\theta \left(f_\theta(I, \text{render}(T)) | \{t_j\}_{j=1}^{k-1} \right), \{t_j\}_{j=1}^{k-1} \right]$$

- Distribution over traces

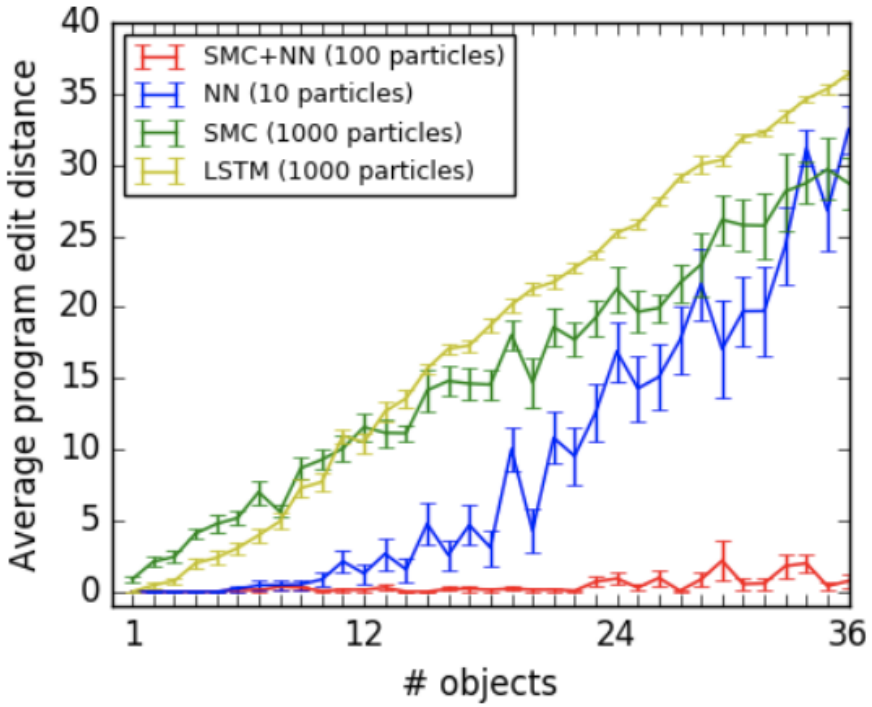
$$\mathbb{P}_\theta[T|I] = \prod_{n=1}^{|T|} \mathbb{P}_\theta[T_n | I, T_{1:(n-1)}] \times \mathbb{P}_\theta[\texttt{STOP} | I, T]$$

TRAINING

- Network may predict incorrectly
- To make the network more robust, SMC is employed
- SMC – Sequential Monte Carlo Sampling uses Pixel wise distance as a surrogate for likelihood function

RESULTS

- Edit Distance = size of the symmetric difference between the ground truth trace set and the set produced by the model.



GENERALIZING TO HAND DRAWINGS

- Actual hand drawings are not used, rather noise is introduced into rendering of training target images
- They are produced in the following ways:
  - Rescaling the image intensity by a factor chosen from [0.5, 1.5]
  - Translating the image by ± 3 pixels chosen uniformly
  - Rendering the LATEX using the pencildraw style

LIKELIHOOD FUNCTION

- L learned

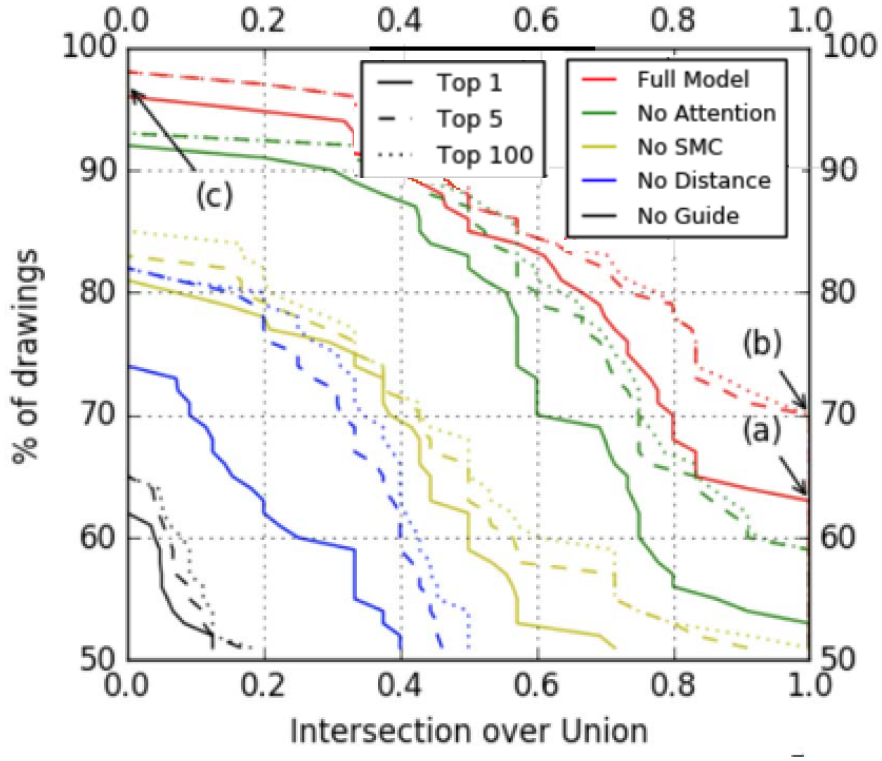
$$-\log L_{\text{learned}}(\text{render}(T_1)|\text{render}(T_2)) \approx |T_1 - T_2| + |T_2 - T_1|$$

- Model now predicts 2 scalars T1 –T2 and T2 - T1

where T1 is rendered output with noise
and T2 is rendered output

EVALUATION

- 100 hand drawn images on graph paper snapped to a 16 x 16 grid
- IoU is intersection over union between ground truth and predicted trace set
- We use IoU to measure the system's accuracy at recovering trace sets.



STAGE 2 : PROGRAM SYNTHESIS

Program→	Statement; ⋯ ; Statement
Statement→	`circle`(Expression,Expression)
Statement→	`rectangle`(Expression,Expression,Expression,Expression)
Statement→	`line`(Expression,Expression,Expression,Expression,Boolean,Boolean)
Statement→	`for`$(0 \leq$ Var $<$ Expression$)$ `{ if` $($Var $> 0)$ `{` Program `};` Program `}`
Statement→	`reflect`(Axis) `{` Program `}`
Expression→	$\mathbb{Z}\times$Var$+\mathbb{Z}$
Var→	A free (unused) variable
$\mathbb{Z}\rightarrow$	an integer
Axis→	`X =` $\mathbb{Z}$ `| Y =` $\mathbb{Z}$

COST FUNCTION

$$\text{program}(T) = \underset{p \in \text{DSL, s.t. } p \text{ evaluates to } T}{\arg\min} \text{cost}(p)$$

- Cost is computed in the following way
  - Programs incur a cost of 1 for each command (primitive drawing action, loop, or reflection)
  - They incur a cost of 1/3 for each unique coefficient they use in a linear transformation

BIAS OPTIMAL SEARCH POLICY

- **Bias Optimality Search Algorithm :** A search algorithm is n-bias optimal w.r.t a distribution $\mathbb{P}_{\text{bias}}[\cdot]$ if it is guaranteed to find a solution in (σ) after searching for at least $n \times \frac{t(\sigma)}{\mathbb{P}_{\text{bias}}[\sigma]}$.
- We use a 1 – bias optimal search algorithm which means we spend $\mathbb{P}_{\text{bias}}[\cdot]$ time looking in the program space.
- This implies that we look in the whole program space

TRAINING

- Policy is given $\pi_\theta(\sigma|T)$
- Define a loss to train the search Policy

$$\text{Loss}(\theta; \mathcal{D}) = \mathbb{E}_{T\sim\mathcal{D}}\left[\min_{\sigma\in\text{BEST}(T)} \frac{t(\sigma|T)}{\pi_\theta(\sigma|T)}\right] + \lambda\|\theta\|_2^2$$

where $\sigma \in \text{BEST}(T)$ if a minimum cost program for T is in σ.

- D is the training corpus i.e. minimum cost programs for few trace sets.
- θ denotes parameters of the search policy

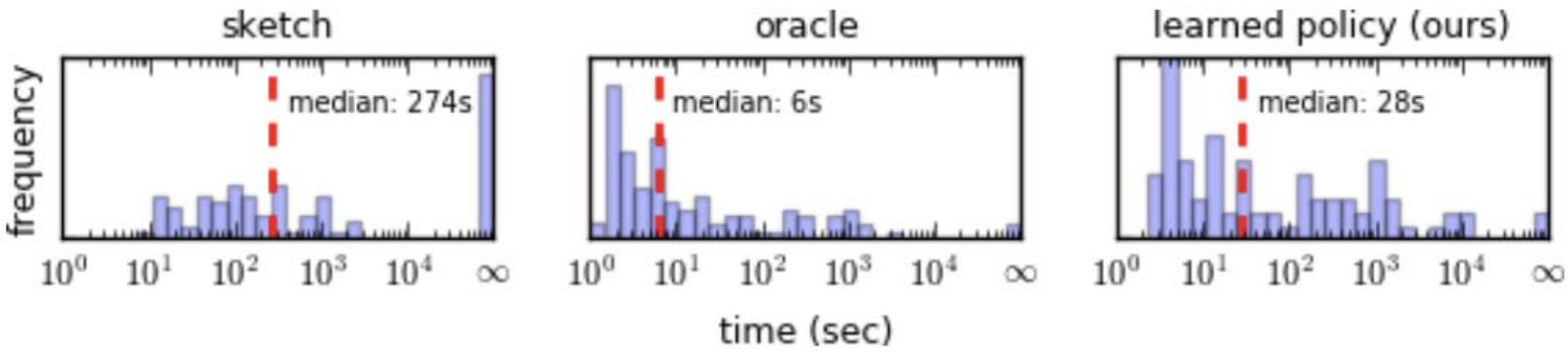
BILINEAR MODEL

$$\pi_\theta(\sigma|T) \propto \exp\left(\phi_{\text{params}}(\sigma)^\top \theta \phi_{\text{trace}}(T)\right)$$

where $\phi_{\text{params}}(\sigma)$ denote a one-hot encoding of the parameter settings of sigma, 24 dimensions long

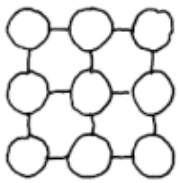
Parameter	Description	Range
Loops?	Is the program allowed to loop?	{True, False}
Reflects?	Is the program allowed to have reflections?	{True, False}
Incremental?	Solve the problem piece-by-piece or all at once?	{True, False}
Maximum depth	Bound on the depth of the program syntax tree	{1, 2, 3}

SEARCH POLICY - COMPARISON



COMPRESSION (TRACE SET AND PROGRAM)

- Ising Model – Lattice Structure



```
Circle(5,8)
Circle(2,8)
Circle(8,11)
Line(2,9, 2,10)
Circle(8,8)
Line(3,8, 4,8)
Line(3,11, 4,11)
```

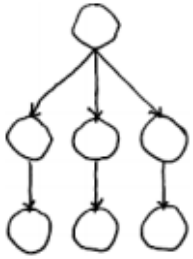
... etc. ...; 21 lines

```
for(i<3)
 for(j<3)
  if(j>0)
   line(-3*j+8,-3*i+7,
        -3*j+9,-3*i+7)
   line(-3*i+7,-3*j+8,
        -3*i+7,-3*j+9)
  circle(-3*j+7,-3*i+7)
```

Compression Factor

21/6 = 3.5x

- Another example



```
Line(5,13,2,10,arrow)
Circle(5,9)
Circle(8,5)
Line(2,8, 2,6,arrow)
Circle(2,5)
```

... etc. ...; 13 lines

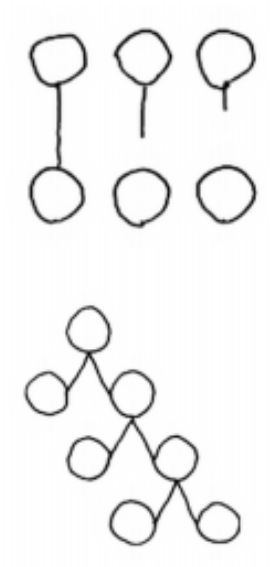
```
circle(4,10)
for(i<3)
 circle(-3*i+7,5)
 circle(-3*i+7,1)
 line(-3*i+7,4,-3*i+7,2,arrow)
 line(4,9,-3*i+7,6,arrow)
```

Compression Factor

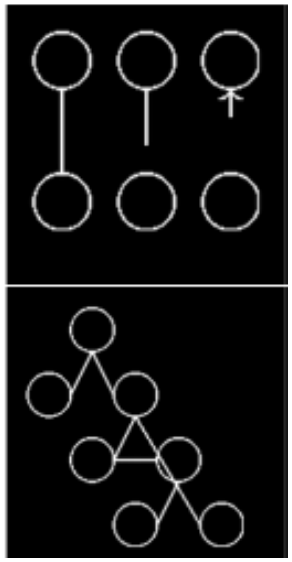
13/6 = 2.2x

APPLICATIONS

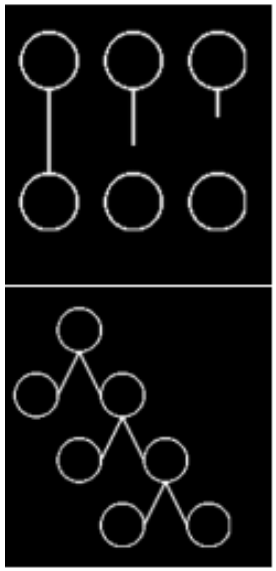
- Correcting Errors made by the neural network using Prior Probability of programs



Hand Drawing



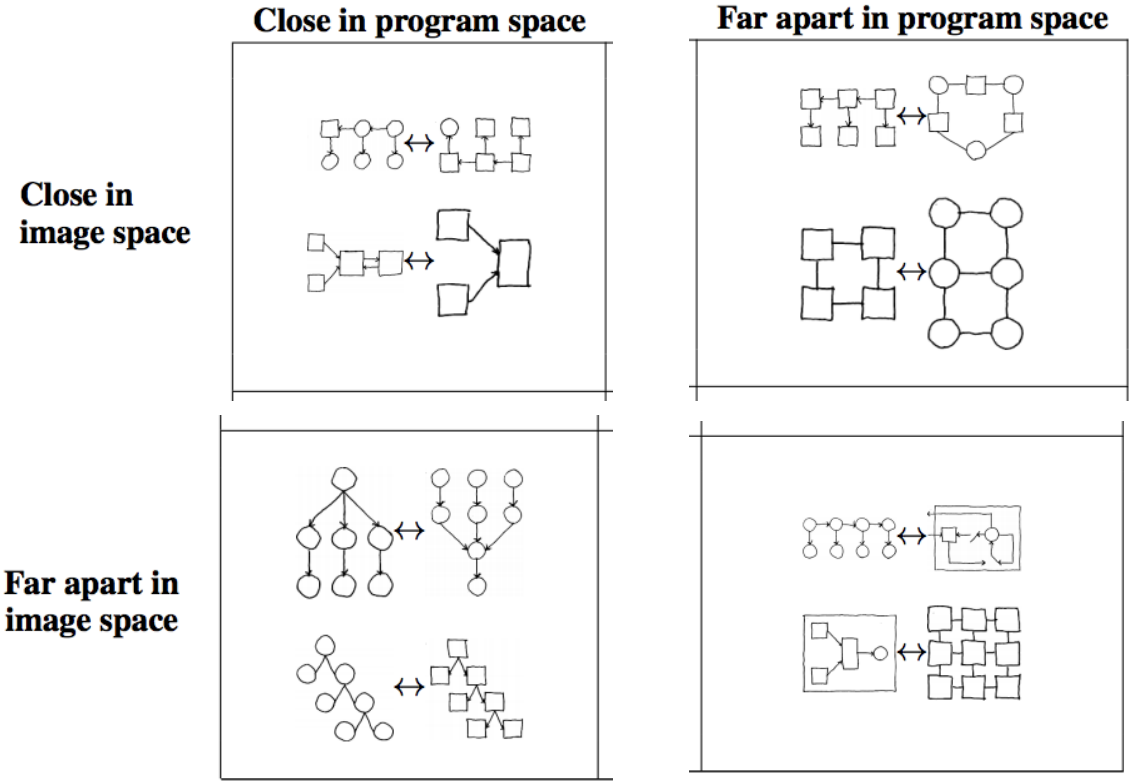
Trace Set – Rendered in Latex



Program Synthesizer's Output

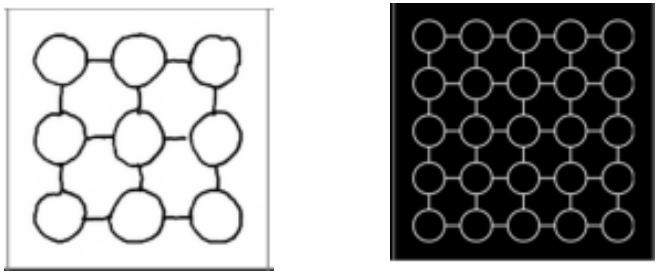
APPLICATIONS

- Modelling similarity between Drawings

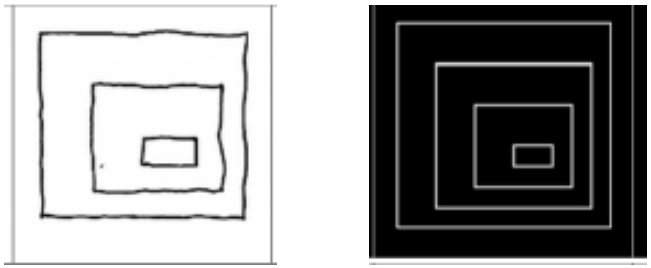


APPLICATIONS

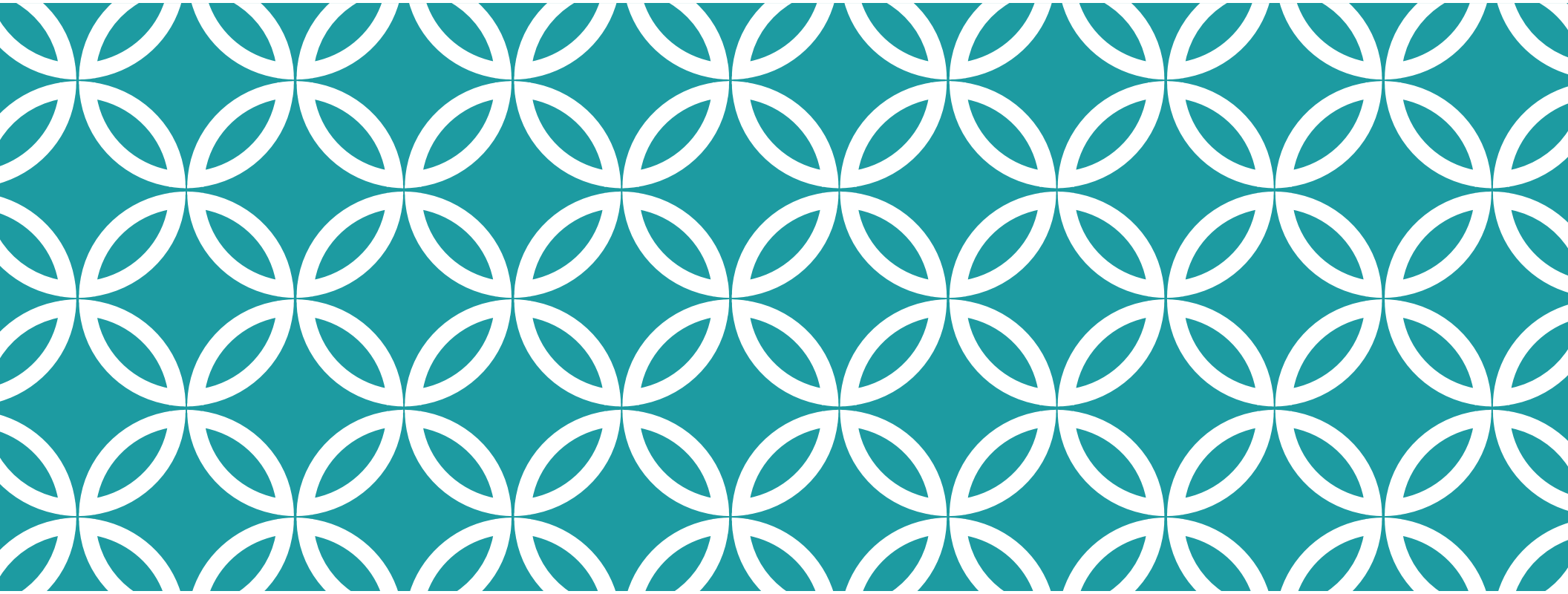
- Extrapolating Figures



Example 1



Example 2



THANK YOU

Discussion