Chapter 1: Introduction to One-shot Learning

![Neuron diagram with labels: Nucleus, Cell body, Axon, Dendrites.]

![Neural network diagram with nodes labeled x1, x2, W1, W2, W3, and sigmoid function leading to f(x).]
Chapter 2: Metrics-Based Methods

The figure shows a neural network architecture with layers labeled as RELU, Convolutional (CONV), and Pooling (POOL). The input images are processed through these layers, resulting in feature maps that are analyzed for classification. The output layer (FC) predicts the class, with examples for car, truck, airplane, ship, and horse.
The Distance Function decides if the output vectors are close enough to be similar.

The Neural Network transforms the input into a properties vector.

Input Data (image, text, features...)

---

Similar/Not?

$\begin{array}{c}
\text{NN} \\
X_2 \\
\text{Same Network} \\
\text{NN} \\
X_1
\end{array}$

---

$x_1$ $x_2$ label

$\begin{array}{c}
\text{च} \text{च} & 1 \\
\text{र} \text{श} & 0 \\
\text{स} \text{स} & 1
\end{array}$

---

Anchor

Positive

Negative

Learning

Anchor

Positive

Negative
Support Set (S)

\[ y_i, x_i \]

\[ g(x_i) \]

\[ f \]

\[ f(\hat{x}, S) \]

\[ a(\hat{x}, x_i) \]

\[ y_i \]

\[ \sum \]

\[ P(\hat{y}|\hat{x}, S) = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \]

\[ \theta = \arg \max_{\theta} E_{L \sim T} \left[ E_{S \sim L, B \sim L} \left[ \sum_{(x,y) \in B} \log P_{\theta}(y|x, S) \right] \right] \]

\[ a(\hat{x}, x_i) = e^{c(f(\hat{x}), g(x_i))} / \sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))} \]

\[ c: \text{cosine distance} \]
Chapter 3: Model-Based Methods
Fast parameterization

\[ \theta^+ = F_w(\nabla_\theta \mathcal{L}^{\text{emb}}_1, \ldots, \nabla_\theta \mathcal{L}^{\text{emb}}_K) \]

Meta learner:

\[ f_{\theta, \theta^+} \]

Input

Base learner:

\[ g_{\phi, \phi^+} \]

Output

Slow weights \( \theta \) | Fast weights \( \theta^+ \)

“Key” memory

\[ R = \{r'_i\}_{i=1}^K \]

Slow weights \( \phi \) | Fast weights \( \phi^+ \)

“Value” Memory

\[ M = \{\phi^+_i\}_{i=1}^K \]

Fast parameterization

\[ \phi^+_i = G_{\phi}(\nabla_\phi \mathcal{L}^{\text{task}}_i) \]

Meta Info

\[ r'_i = f_{\theta, \theta'}(x'_i) \]
Chapter 4: Optimization-Based Methods

\[ \theta \]
\[ \nabla \mathcal{L}_1 \]
\[ \nabla \mathcal{L}_2 \]
\[ \nabla \mathcal{L}_3 \]
\[ \theta_1^* \]
\[ \theta_2^* \]
\[ \theta_3^* \]

---

**Domain-Adaptive Meta-Learning**

- Provide demonstration data

- Human demos

- Robot demos

- Learn how to infer a policy from one human demo

---

**Deployment**

- Provide one video of human

- Infer robot policy

---

[14]
(X_i, Y_i) : mini-batches sampled from D^{(d)}_{train}

(X_1, Y_1) \rightarrow \theta_0 \rightarrow M_{\theta_0} \rightarrow R_{\Theta_d} \rightarrow (\nabla_1, \mathcal{L}_1)

(X_2, Y_2) \rightarrow \theta_1 \rightarrow M_{\theta_1} \rightarrow R_{\Theta_d} \rightarrow (\nabla_2, \mathcal{L}_2)

\ldots

(X_T, Y_T) \rightarrow \theta_{T-1} \rightarrow M_{\theta_{T-1}} \rightarrow R_{\Theta_d} \rightarrow (\nabla_T, \mathcal{L}_T)

\rightarrow \theta_T \rightarrow M_{\theta_T} \rightarrow \mathcal{L}(M_{\theta_T}(X), Y) \rightarrow \rightarrow R_{\Theta_{d+1}}

\text{LSTM}

\text{Learner} : \text{outputs the learner's parameter}

Repeats T steps; t = 1, ..., D

Repeats D steps; d = 1, ..., D

Meta-learner : outputs the learner's parameter
Chapter 5: Generative Modeling-Based Methods

![Diagram of generative modeling-based methods]

- Models that think there are more right-handed people
- Models that think there are more left-handed people

```
P(θ|Data) vs. Models (θ)
```

```
(Λ) \rightarrow \{θ_{MAP}\} MAP_{estimate}
```
<table>
<thead>
<tr>
<th>B</th>
<th>i)</th>
<th>iii)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>

| ii) | ![Image](image3.png) |
| iv) | ![Image](image4.png) | ![Image](image5.png) |
procedure `generateType`

\[ \kappa \leftarrow P(\kappa) \]  
\[ \text{for } i = 1 \ldots \kappa \text{ do} \]  
\[ n_i \leftarrow P(n_i | \kappa) \]  
\[ \text{for } j = 1 \ldots n_i \text{ do} \]  
\[ s_{ij} \leftarrow P(s_{ij} | s_{(j-1)i}) \]  
\[ R_i \leftarrow P(R_i | S_i, \ldots, S_{i-1}) \]  
\[ \psi \leftarrow \{e, R, S\} \]  
\[ \text{return } @\text{generateToken}(\psi) \]  

procedure `generateToken(\psi)`

\[ S_i^{(m)} \leftarrow P(S_i^{(m)} | S_i) \]  
\[ L_i^{(m)} \leftarrow P(L_i^{(m)} | R_i, T_i^{(m)} \ldots, T_i^{(m)}) \]  
\[ T_i^{(m)} \leftarrow f(L_i^{(m)}, S_i^{(m)}) \]  
\[ \text{for } i = 1 \ldots N \text{ do} \]  
\[ A_i^{(m)} \leftarrow P(A_i^{(m)}) \]  
\[ f(m) \leftarrow P(f(m) | T(m), A(m)) \]  
\[ \text{return } f(m) \]  

phase 2: concept learning

\[ \bar{W} \]

phase 1: representational learning

\[ \widetilde{y}_i \]
\[ \widetilde{x}_i \]
\[ \bar{\mathcal{D}} = \{\widetilde{x}_i, \widetilde{y}_i\}_{i=1}^{N} \]

phase 3: k-shot learning

\[ y_i \]
\[ x_i \]
\[ \mathcal{D} = \{x_i, y_i\}_{i=1}^{N} \]

phase 4: k-shot testing

\[ \bar{y}^* \]
\[ \bar{x}^* \]
Chapter 6: Conclusions and Other Approaches