Learning to Stop While Learning to Predict (ICML 2020)

Author: Xinshi Chen, Hanjun Dai, Yu Li, Xin Gao, Le Song

Presenter: Linlin Yu

March 12, 2021

Outline

Paper Related

- Paper Address
- Authors' slide
- ICML Poster
- Code

Background Material

- Deep Reinforcement Learning: Stanford CS231n Lecture 14
- KL Divergence:
  - Dibya Ghosh's blog
  - Agustinus Kristiadi's blog
- Variational Autoencoder(VAE)
  - Part of Stanford cs231n Lecture 13
  - Jeremy Jordan's blog
  - If you want to go deep in this area, there are some materials.
    - One of the most famouns paper: Auto-Encoding Variational Bayes
    - An introduction from Max Welling: An Introduction to Variational Autoencoders
    - Chapter 10 in PRML book from Bishop.

KL Divergence

- It is used to measure the difference between two distributions.(Kullback-Leibler Divergence)

$$KL(p(x)||q(x)) = \int p(x) \log \frac{p(x)}{q(x)} dx = \mathbb{E}_{x \sim p(x)}(\log \frac{p(x)}{q(x)})$$

- It can be called relative entropy since:

$$KL(p(x)||q(x)) = \int p(x) \log p(x) dx - \int p(x) \log q(x) dx$$

- It has been widely used in machine learning and information theory.

Features of KL

- Asymmetric: $KL(p||q) \neq KL(q||p)$
- $KL(p(x)||q(x)) \geq 0$ and it is equal to 0 when $p(x) = q(x)$
- Infinite Problem: When $q(x)$ is equal to 0 in some range and $p(x) \neq 0$, then we have KL tend to infinite
- Advantage: it can be written as an expectation and we can use sampling to approximate

Forward KL VS Reverse KL

- Forward KL (Mean-Seeking Behaviour)

$$\begin{aligned} \underset{\theta}{argmin}\, KL(P||Q) &= \underset{\theta}{argmin}\left(\int p(x) \ln \frac{p(x)}{q_\theta(x)} dx\right) \\ &= \underset{\theta}{argmax}\, \mathbb{E}_{x\sim p(x)}[\ln q_\theta(x)] \end{aligned}$$

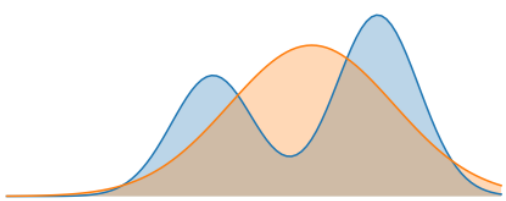
It is identical to the MLE. **Wherever P has high probability and Q must also have high probability.**

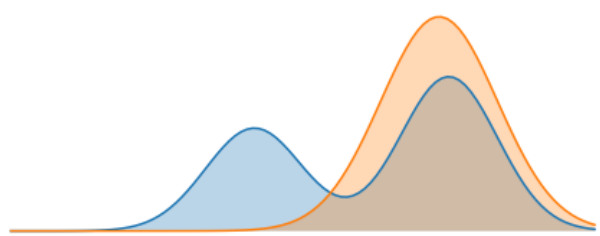
Zero Avoiding: It avoids $q(x) = 0$ when $p(x) > 0$, since $p(x) \ln \frac{p(x)}{q_\theta(x)}$

- Reverse KL (Mode-Seeking Behaviour)

$$\begin{aligned} \underset{\theta}{argmin}\, KL(Q||P) &= \underset{\theta}{argmin}\left(\int q_\theta(x) \ln \frac{q_\theta(x)}{p(x)} dx\right) \\ &= \underset{\theta}{argmax}(\mathbb{E}_{x\sim q_\theta(x)}[\ln p(x)] + H[q_\theta(x)]) \end{aligned}$$

Wherever Q has high probability and P must also have high probability.
zero forcing: It forces $q(x)$ to be zero on some areas even if $p(x) > 0$(no penalty for these situations), which means to try to avoid spreading the approximate





Forward KL VS Reverse KL

- The blue contours show a bimodal distribution $p(\mathbf{X})$ given by a mixture of two Gaussians;
- The red contours correspond to the single Gaussian distribution $q(\mathbf{X})$ that best approximates $p(\mathbf{X})$ in the sense of minimizing the KL divergence.
  - Figure (a): Forward KL, which means minimizing $KL(p\|q)$
  - Figure (b): Reverse KL, which means minimizing $KL(q\|p)$
  - Figure (c): Reverse KL, similar to (b) but showing a different local minimum of the $KL(q\|p)$

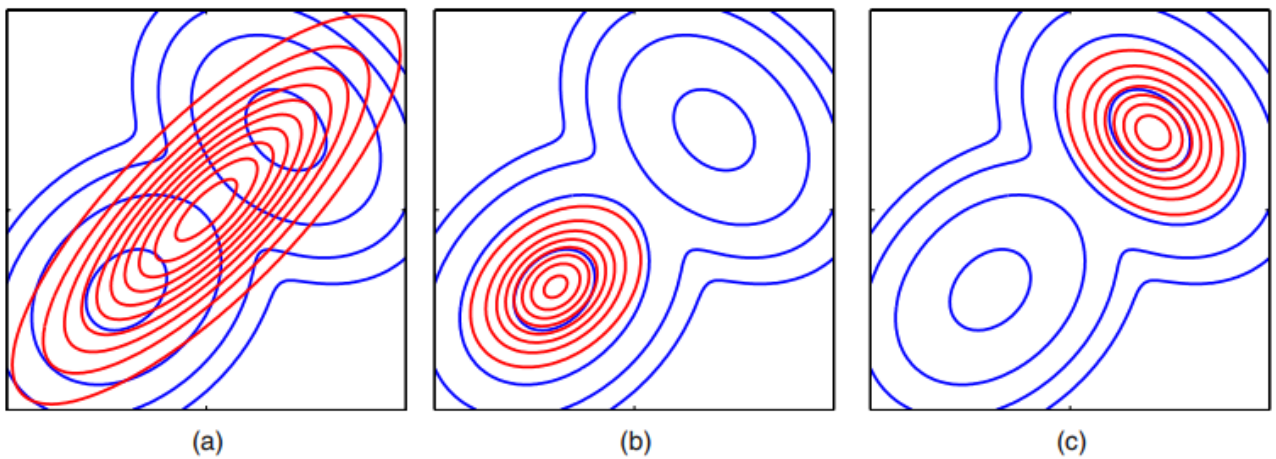


Figure: 10.2 in PRML book

How to Choose

1. Forward KL: average across all of the modes;
2. Reverse KL: may only capture one mode

- It depends. For example, reverse KL is widely used in varational inference since it is used in the original paper. However, it has a mode collapse issue.
- In this paper, the author use forward KL and has a better performance.The reason is that the mode collapse issue can make the model only captures a common stopping time.
- Mode Collapse Issue: It is a common issue in GAN technology, where the generator fails to capture all existing modes of the input distribution. For example, a generator makes multiple images that contain the same color or texture themes or multiple images containing different views of the same dog.

Forward KL VS Reverse KL

Forward KL is equal to a supervised learning:

- For supervised learning, we need to do the follows:

$$\underset{\theta}{argmin}\, \mathbb{E}_{(x,y)\sim \mathrm{D}}[L(f_\theta(x), y)]$$

- Minimizing the KL divergence $KL(p(y|x)||q_\theta(y|x))$ needs to do follows:

$$\underset{\theta}{argmin}\, \mathbb{E}_{(x,y)\sim \mathrm{D}}[-\ln Q_\theta(y|x)]$$

1. For classification, the entropy loss is just as this form;
2. For Regression, we assume that $q_\theta(y|x) \sim \mathcal{N}(f_\theta(x), I)$,then we have:

$$q_\theta(y|x) = \frac{1}{\sqrt{2\pi}} \cdot e^{\frac{(y-f_\theta(x))^2}{2}}$$

Therefore,

$$-\log q_\theta(y|x) = -\frac{1}{2}(y - f_\theta(x))^2 - \ln \frac{1}{\sqrt{2\pi}}$$

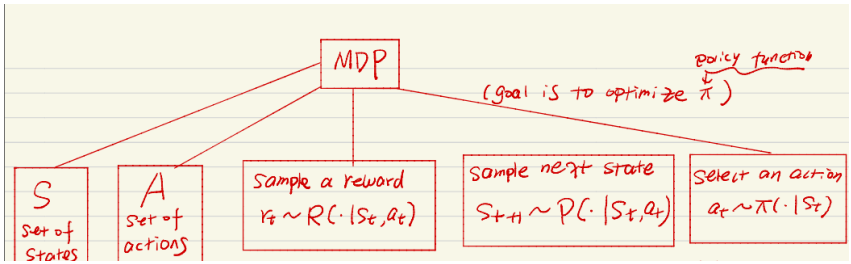
Minimizing the NLL of this normal distribution is clearly equivalent to the mean-squared error loss

- This is the first condition of **Lemma 1**

Forward KL VS Reverse KL

Reverse KL is equal to a maximum-entropy Reinforcement learning

Consider a MDP Markov Decision Process(from Dr.Chen's AI course):



A trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$ and $P(\tau) \propto \prod_{t=0}^{T-1} p(a_t|s_t)p(s_{t+1}|s_t, a_t)$

RL objective:

$$\pi^* = \underset{\pi}{argmax}\, \mathbb{E}_\pi[\sum_t R(s_t, a_t)]$$

Maximum-entropy optimization objective [4]

$$\pi^* = \arg\max_\pi \mathbb{E}_\pi \left[\sum_{t=1}^{T} \underbrace{R(s_t, a_t)}_{\text{reward}} + \alpha \underbrace{\mathrm{H}(\pi(\cdot \mid s_t))}_{\text{entropy}} \right]$$

In detail, it is ($\alpha = 1$)

$$\underset{\pi}{argmax}\, \mathbb{E}_{\tau \sim q(\pi)}[\sum_{t=1}^{T}(r(s_t, a_t) - \log \pi(a_t|s_t))]$$

Forward KL VS Reverse KL

Assume: $P_{opt}(\tau)$: a distribution of optimal trajectories;
Goal: Learn policy $\pi(a|s)$ which induces a distribution over trajectories $q_\pi(\tau)$
Given: the probability of a trajectory under optimality is exponential in the sum of rewards received on the trajectory

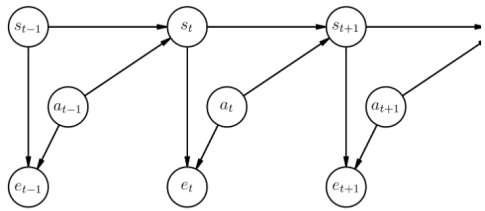
$$\log P(\tau) = \sum_{t=1}^{T} r(s_t, a_t)$$

Optimizing the reverse KL objective between $P_{opt}(\tau)$ and $q_\pi(\tau)$:

$$\begin{aligned}
\underset{\pi}{argmax}\, KL(q_\pi \| P(\tau)) &= \underset{\pi}{argmax}\, \mathbb{E}_{\tau\sim q(\pi)}[\log(P(\tau)] + H(q_\pi(\tau)) \\
&= \underset{\pi}{argmax}\, \mathbb{E}_{\tau\sim q(\pi)}[\sum_{t=1}^{T} r(s_t, a_t)] + \mathbb{E}_{\tau\sim q(\pi)}[\sum_{t=1}^{T} -\log \pi(a_t|s_t)] \\
&= \underset{\pi}{argmax}\, \mathbb{E}_{\tau\sim q(\pi)}[\sum_{t=1}^{T} (r(s_t, a_t) - \log \pi(a_t|s_t))]
\end{aligned}$$

Deep understanding of the mapping [5]

Model the trajectories into a graphical model:



where e_t is a binary random variable at every timestep, which indicates whether it is a optimal decision or not. $P(e_t = 1|s_t, a_t) = f(s_t, a_t)$ and let $r(s_t, a_t) = \log f(s_t, a_t)$

For a trajectory τ, the probability that it is optimal at all timesteps $P(all \quad e_t = 1|\tau) = \exp(\sum_{t=0}^{T} r(s_t, a_t))$

Then the optimal trajectory distribution conditioned on being optimal all time steps:

$\pi_{optimal}(\tau) = P(\tau|all \quad e_t = 1) \propto exp(\sum_{t=0}^{T} r(s_t, a_t))P(\tau)$ (Bayes rule)

Target: to learn a variational distribution $q_\theta(\tau)$ to approximate $P(\tau|all \quad e_t = 1)$.

$$q_\theta(\tau) = (\prod_{t=0}^{T} q(a_t|s_t))P(\tau)$$

Then we can derive the maximum entropy loss from the reverse KL divergence.

A simple introduction of Variational Inference Auto-encoder(VAE)

Target: Use latent variable to describe the distribution of the input data

Assume: the input data follows $\widetilde{p}(x)$ and a latent variable z

The joint distribution of x and the latent variable z is

$$p(x, z) = \widetilde{p}(x) \cdot p(z|x)$$

In fact we don't know the input data distribution and we use $q(x)$ to describe it (we want to learn $q(x)$)

$$q(x) = \int q(x|z) \cdot q(z) dz$$

- $q(z)$ is the prior probability of latent variable z
- $q(x|z)$ denotes the ability of the generation (we use latent variable to generate data like the input data)

Derivation of Loss of VAE

Training Target: We want use $q(x)$ to approximate $\widetilde{p}(x)$. In practice, we use the joint distribution to approximate.

$$
\begin{aligned}
KL(p(x,z)\|q(x,z)) &= \int\int p(x,z)\ln\frac{p(x,z)}{q(x,z)}dzdx \\
&= \int \widetilde{p}(x)\int p(z|x)\ln\frac{\widetilde{p}(x)p(z|x)}{q(x,z)} \\
&= \mathbb{E}_{x\sim\widetilde{p}(x)}[\int p(z|x)\ln\frac{\widetilde{p}(x)p(z|x)}{q(x,z)}] \\
&= \mathbb{E}_{x\sim\widetilde{p}(x)}[\ln\widetilde{p}(x)\int p(z|x)dz + \int p(z|x)\ln\frac{p(z|x)}{q(x,z)}dz] \\
&= const + \mathbb{E}_{x\sim\widetilde{p}(x)}\int p(z|x)\ln\frac{p(z|x)}{q(x,z)}dz
\end{aligned}
$$

$\mathcal{L} = KL(p(x,z)\|q(x,z)) - const$ and $const = \mathbb{E}_{x\sim\widetilde{p}(x)}[\ln\widetilde{p}(x)]$

Loss of VAE

Loss of VAE:

$$\mathcal{L} = \mathbb{E}_{x\sim\tilde{p}(x)}[]\mathbb{E}_{z\sim p(z|x)}[-\log q(x|z)] + KL(p(z|x)||q(z))]$$

What we need: posterior: $p(z|x)$, prior: $q(z)$, and likelihood $q(x|z)$
What we do:maximize the probability of generated data (the data likelihood under the posterior), while keeping the distance between the posterior and prior latent distributions small

A classic model of VAE [2]

Suppose the prior $p(z) \sim N(0, I)$. Then we use network to fit the posterior(encoder) and likilihood (decoder).
Actually, we assume that the posterior is also a Gaussian Distribution and we get the mean and variance from the encoder.

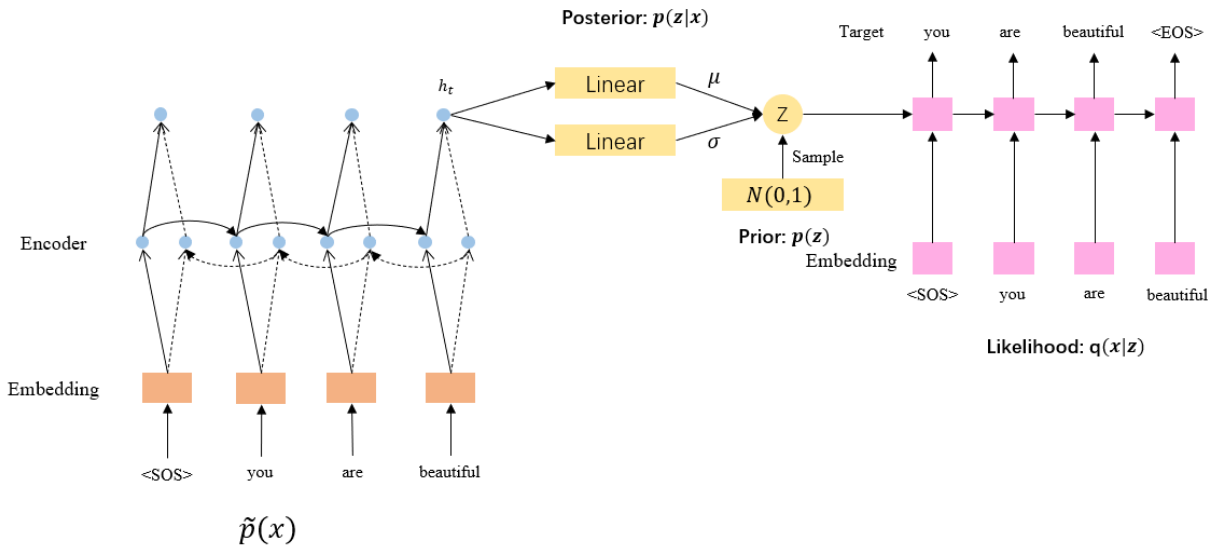


Figure: Use neural network to learn

$\beta - VAE$

Loss of $\beta - VAE$:

$$\mathcal{L} = E_{z\sim p(z|x)}[-\log q(x|z)] + \beta \dot{K} L(p(z|x)||q(z))$$

Advantage of $\beta - VAE$
When $\beta = 1$, it is a standard VAE. If $\beta > 1$, it applies a stronger constraint on the latent bottleneck and limits the representation capacity of z. Therefore a higher β encourages more efficient latent encoding and further encourages the disentanglement. Meanwhile, a higher β may create a trade-off between reconstruction quality and the extent of disentanglement
What is disentanglement: For example, a model trained on photos of human faces might capture the gentle, skin color, hair color, hair length, emotion, whether wearing a pair of glasses and many other relatively independent factors in separate dimensions. Such a disentangled representation is very beneficial to facial image generation.

Motivation

1. Data-Driven Algorithm Design

Unrolled(unfolded) Algorithm: Given that deep network lacks interpretability and need for a very large training sets, unrolling algorithm provides a concrete and systematic connection between traditional iterative algorithms and deep neural networks.
Application: Sparse coding(first proposed), parameter learning in graphical models, structured prediction....

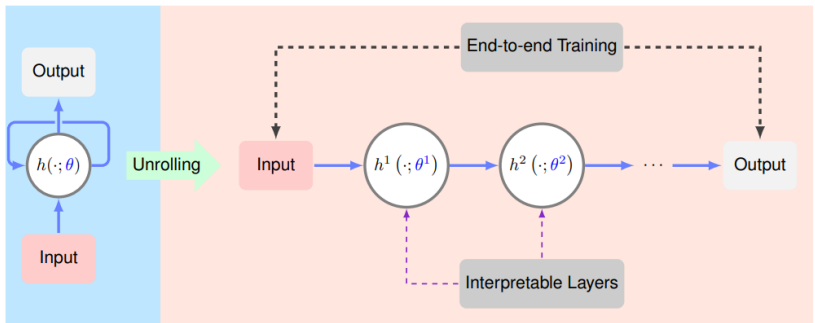


Figure: A high-level overview of algorithm unrolling [6]

Motivation

1. Data-Driven Algorithm Design

Traditional algorithms have certain stop criteria to determine the number of iterations for each problem. E.g.,

- iterate until convergence
- early stopping to avoid over-fitting

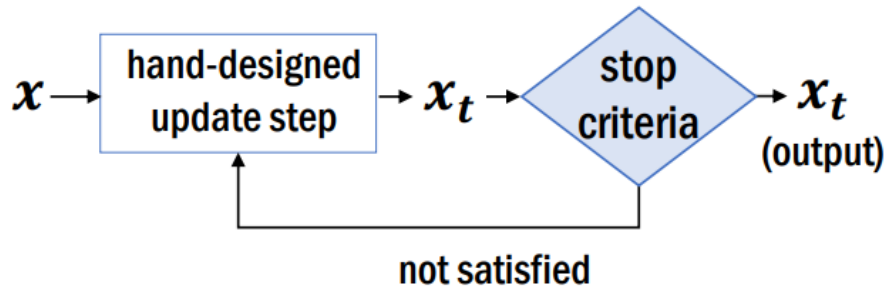


Figure: Dynamic-depth Traditional Algorithm

Current work usually have a **fixed number of iterations** in the architecture.
A new view to consider "**stop**" from the relation between deep learning models and traditional algorithms

Motivation

2. Task-imbalanced Meta Learning

Meta Learning:Learn to Learn

Let the machine to be a better learner from experience, which refers to improving a learning algorithm over multiple learning episodes

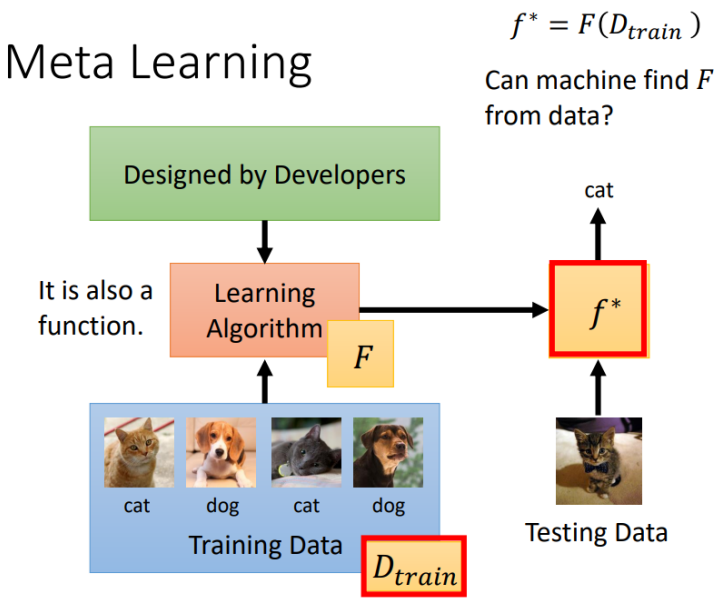


Figure: Adapted from [7]

2.Task-imbalanced Meta Learning

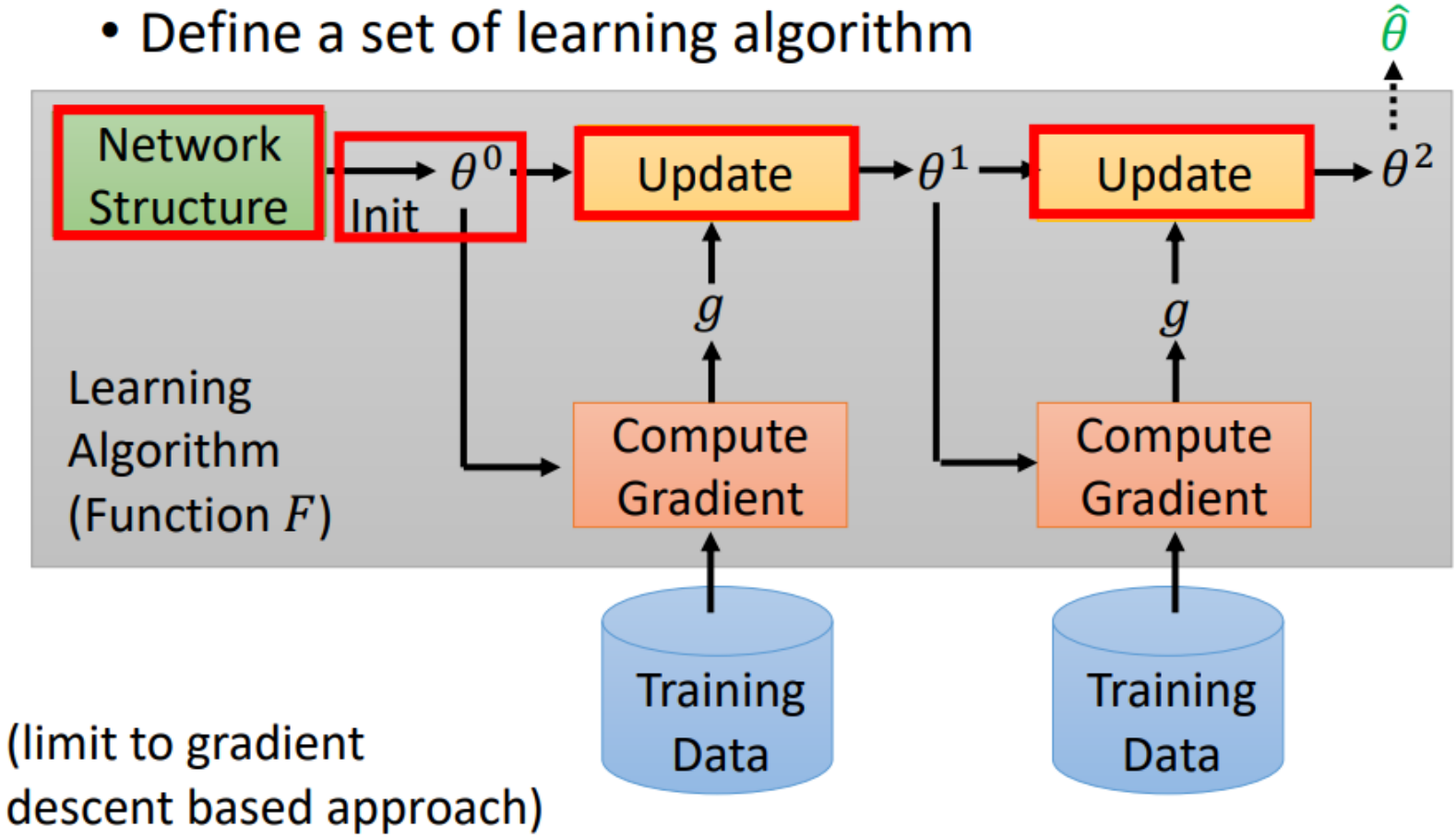


Figure: Adapted from [7]

2.Task-imbalanced Meta Learning

Meta Learning $L(F) = \sum_{n=1}^{N} l^n$ (N: N tasks; l^n: Testing loss for task n after training)

- Defining the goodness of a function F

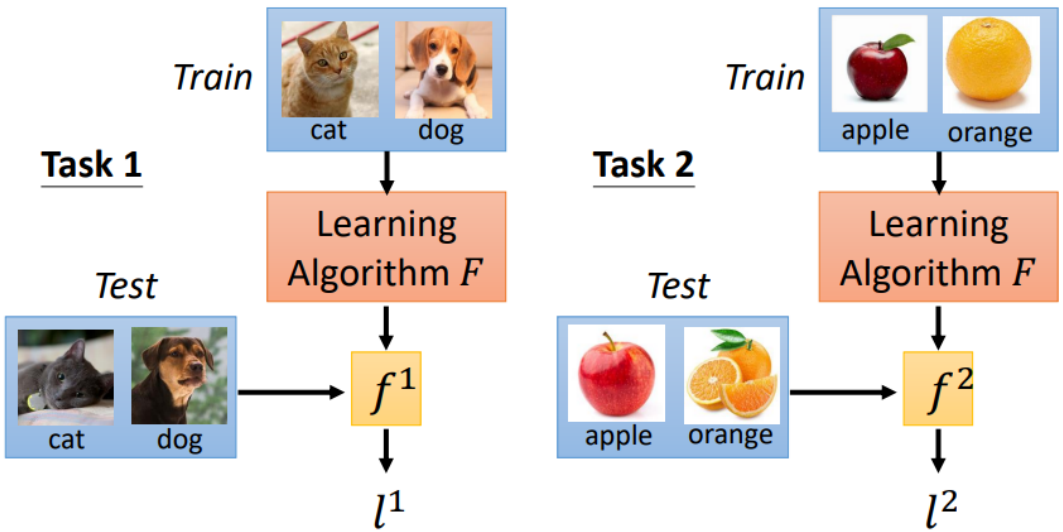


Figure: Training Stage: Adapted from [7]

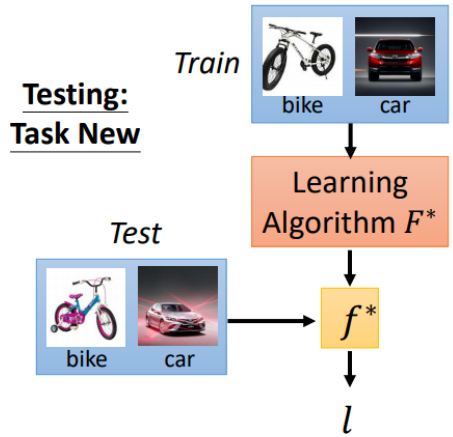


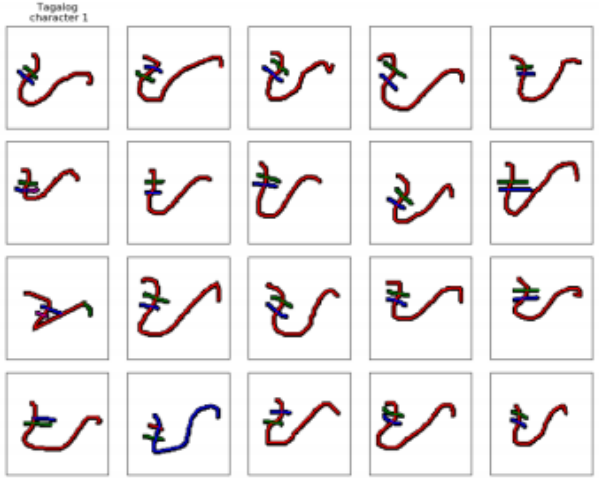
Figure: Testing Stage: Adapted from [7]

2.Task-imbalanced Meta Learning

Omniglot

https://github.com/brendenlake/omniglot

- 1623 characters
- Each has 20 examples



2.Task-imbalanced Meta Learning

Omniglot – Few-shot Classification

Demo of Reptile:
https://openai.com/blog/reptile/

- N-ways K-shot classification: In each training and test tasks, there are N classes, each has K examples.

20 ways
1 shot

Each character represents a class

Testing set (Query set)

Training set (Support set)

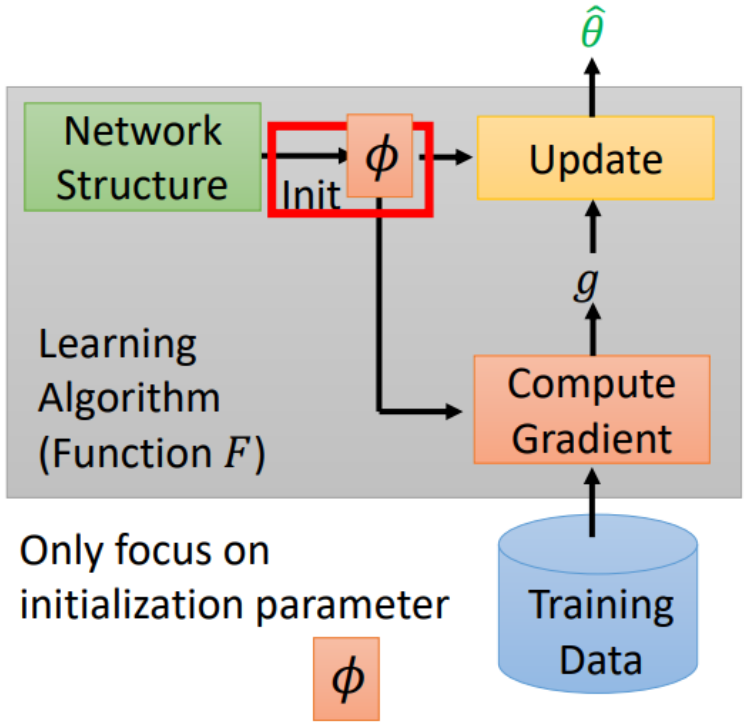
- Split your characters into training and testing characters
  - Sample N training characters, sample K examples from each sampled characters → one training task
  - Sample N testing characters, sample K examples from each sampled characters → one testing task

2.Task-imbalanced Meta Learning

MAML: Model-Agnostic Meta-Learning [3] is an optimization based method. Generally speaking, It trains for a representation that can be adapted to a new task quickly (in a small number of steps) and efficiently (using only a few examples);

MAML

- Fast ... Fast ... Fast ...
- Good to truly train a model with one step. ☺
- When using the algorithm, still update many times.
- Few-shot learning has limited data.



$$L(\phi) = \sum_{n=1}^{N} l^n(\hat{\theta}^n)$$

$$\phi \leftarrow \phi - \eta \nabla_{\phi} L(\phi)$$

Considering one-step training:

$$\hat{\theta} = \phi - \varepsilon \nabla_{\phi} l(\phi)$$

2.Task-imbalanced Meta Learning

Key idea:

- During the course of meta-learning (the bold line), MAML optimizes for a set of parameters such that when a gradient step is taken with respect to a particular task i (the gray lines), the parameters are close to the optimal parameters θ_i for task i

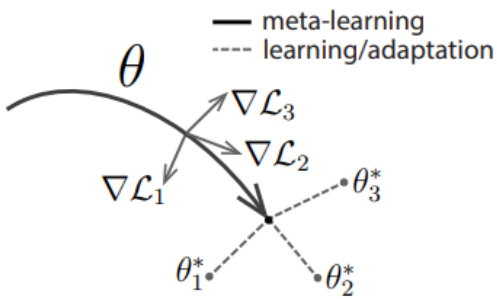


Figure: MAML

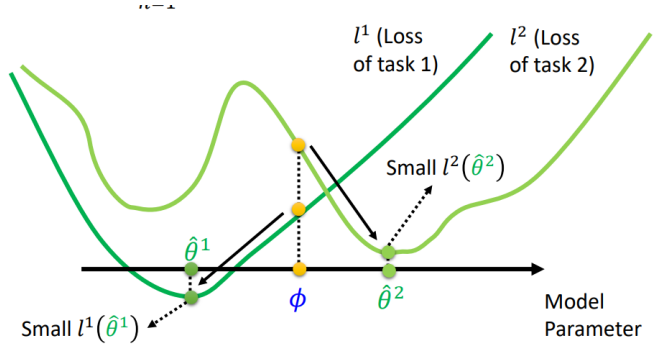
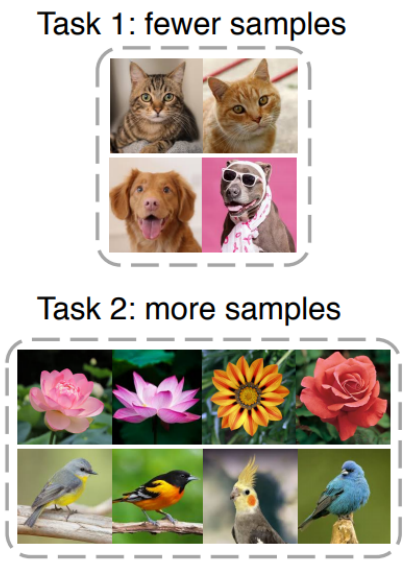


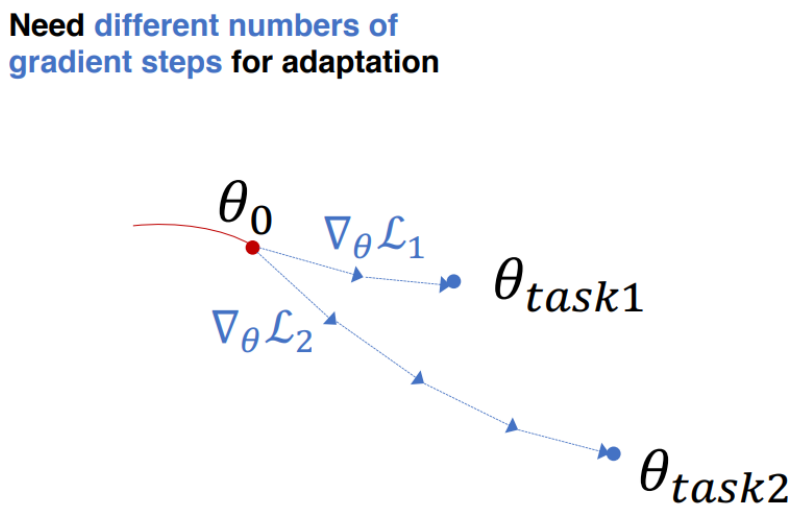
Figure: Training Stage: Adapted from [7]

Motivation

2. Task-imbalanced Meta Learning

A task specified number of adaption steps is more favorable to avoid under or over adaption.





Motivation

3. Image Task

Image Denoising

- Images with different noise levels may need different number of denoising steps

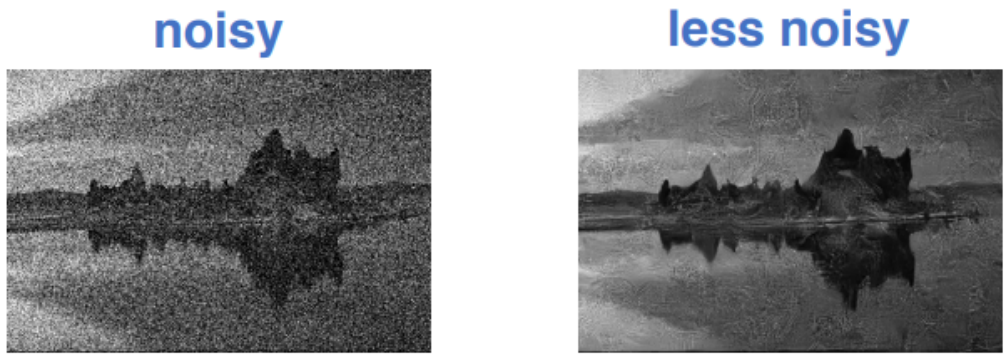


Image Recognition

- 'early exits' is proposed to improve the computation efficiency and avoid 'over-thinking'.

Motivation

4.Optimal Stopping

Stopping rule problems are associated with two objects:

1. A sequence of random variables $X_1, X_2, \ldots$, whose joint distribution is something assumed to be known
2. A sequence of 'reward' functions $(y_i)_{i \geq 1}$ which depend on the observed values of the random variables in 1:

$$y_i = y_i(x_1, \ldots, x_i)$$

Given those objects, the problem is as follows:

- You are observing the sequence of random variables, and at each step i, you can choose to either stop observing or continue
- Option trading: getting back to earlier states is not allowed.
- If you stop observing at step i, you will receive reward y_i
- You want to choose a stopping rule to maximize your expected reward (or equivalently, minimize your expected loss)

Use RL to solve [1]: decomposing an optimal stopping time into a sequence of 0-1 stopping decisions and approximating them recursively with a sequence of multilayer feedforward neural networks.

Predictive Model with Stopping Policy

Predictive model $\mathcal{F}_\theta$

- Transforms the input $\mathbf{x}$ to generate a path of states $\mathbf{x}_1, \cdots, \mathbf{x}_T$

Stopping Policy π_ϕ: $(\mathbf{x}, \mathbf{x}_t) \rightarrow \pi_t \in [0, 1]$

- Sequentially observes the states $\mathbf{x}_t$ and determines the probability of stop at layer t

Variational stop time distribution q_ϕ

- Stop time distribution induced by stopping policy π_ϕ

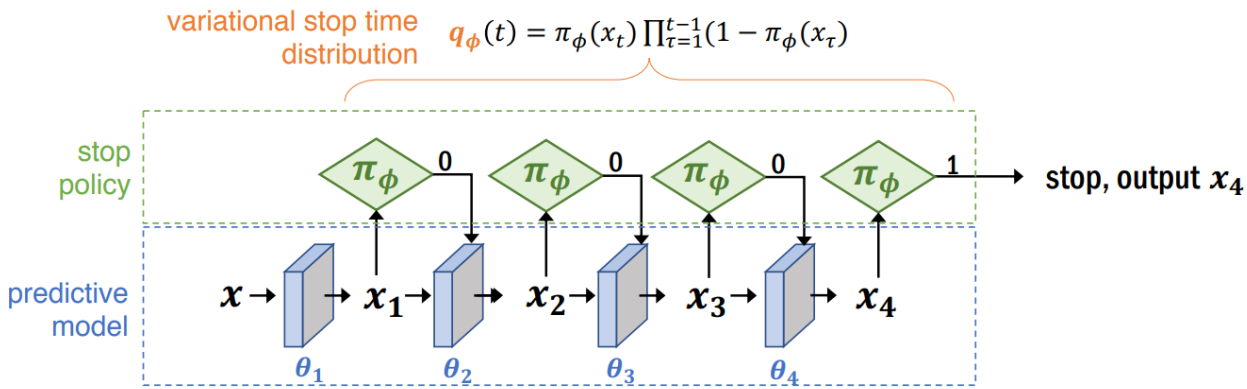


Figure: Two-component model: learning to predict(blue)while learning to stopping(green)

How to learn the optimal $(\mathcal{F}_\theta, \pi_\phi)$ efficiently?

- Design a **joint training objective:**

$$\mathcal{L}(\mathcal{F}_\theta, \mathbf{q}_\phi)$$

- Introduce an oracle stop time distribution:

$$\mathbf{q}^*|\mathcal{F}_\theta := argmin_{a \in \triangle^{T-1}} \mathcal{L}(\mathcal{F}_\theta, \mathbf{q})$$

- Then we decompose the learning procedure into **two stages**:

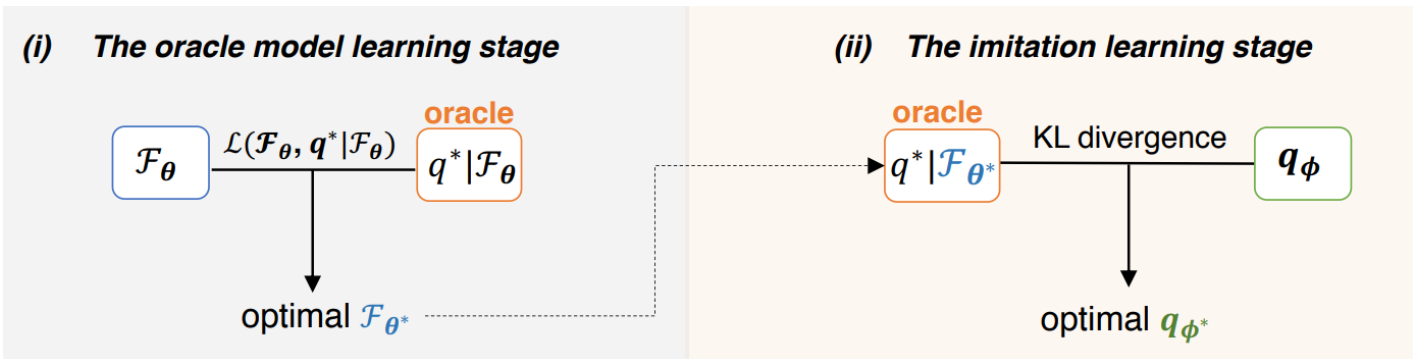


Figure: Two-stage training framework

Problem Formulation

Predictive model $\mathcal{F}_\theta$

A typical T-layer deep model and generates a path of embeddings $(\mathbf{x}_1, \cdots, \mathbf{x}_T)$

$$\mathbf{x}_t = f_{\theta_t}(\mathbf{x}_{t-1}) \tag{1}$$

- $\mathbf{x}_0$ is the input $\mathbf{x}$
- Important difference on calculating the loss:
  - Standard Supervised learning uses the final state $\mathbf{x}_T$ to calculate the loss
  - SDN-training applies a weighted sum of loss functions over all internal classifiers and the final loss function.

$$\min_{\theta_{IC}} \mathcal{L} = \min_{\theta_{IC}} \sum_{i=1}^{7} \tau_i \mathcal{L}_i$$

  - This model uses the expectation loss of all layers according to stop distribution.

$$\mathcal{L}(\theta, q_\phi; \mathbf{x}, \mathbf{y}) := \mathbb{E}_{t \sim q_\phi} \ell(\mathbf{y}, \mathbf{x}_t; \theta) - \beta H(q_\phi) \tag{4}$$

Problem Formulation

Stopping Policy π_ϕ

π_ϕ determines whether to stop at $t-th$ step after observing the input $\mathbf{x}$ and its first t states $\mathbf{x}_{1:t}$. Actually, this model only use the most recent state $\mathbf{x}_t$

$$\pi_t = \pi_\phi(\mathbf{x}, \mathbf{x}_t), for \quad t = 1, \cdots, T-1 \tag{2}$$

- $\pi_t \in [0, 1]$ is the probability of stopping at time step t, π_ϕ is the probability policy and ϕ is the parameter to learn(abusing for simple notation)
- Whether to stop before t is independent on state after t, and once it decides to stop at t, the remaining computations can be saved, if not, then forward the embedding to the subsequent network.

Problem Formulation

Variational Stop time distribution q_ϕ

1. $\pi_t \in [0, 1]$ is the probability of stopping at time step t
2. The product $\prod_{\tau=1}^{t-1}(1 - \pi_\tau)$ indicates the probability of "not stopping before t"
3. Multipy them , we have the stop time distribution $q_\phi(t)$
4. For the last time step T, if the process "not stopped before T", then it must stop at T

$$\begin{cases} q_\phi(t) = \pi_t \prod_{\tau=1}^{t-1}(1 - \pi_\tau) & t < T \\ q_\phi(T) = \prod_{\tau=1}^{T-1}(1 - \pi_\tau) & t = T \end{cases} \tag{3}$$

What we get from the network is π_ϕ and ϕ is the learning parameter.

Problem Formulation

Optimization Objective

Given the observed label $\mathbf{y}$ of an input $\mathbf{x}$, the loss of the predictive model stopped at time t is $\ell(\mathbf{y}, \mathbf{x}_t; \theta)$
Taking into all possible time steps, the overall loss is the loss in expectation over t

$$\mathcal{L}(\theta, q_\phi; \mathbf{x}, \mathbf{y}) := \mathbb{E}_{t \sim q_\phi} \ell(\mathbf{y}, \mathbf{x}_t; \theta) - \beta H(q_\phi) \tag{4}$$

- $H(q_\phi) = -\sum_t q_\phi(t) \log q_\phi(t)$: Entropy regularization
- β: regularization coefficient

$$min_{\theta,\phi} \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \mathcal{L}(\theta, q_\phi; \mathbf{x}, \mathbf{y}) \tag{5}$$

Now, let us use **Variational Bayes** to explain it formally!

Variational Bayes Perspective

- Latent variable t
- Prior: $p(t|\mathbf{x})$: Uniform distribution (change this if you want to penalize the stopping decisions on deeper layer) $\rightarrow$ regularization
- Posterior: $p_\theta(t|\mathbf{y},\mathbf{x}) \rightarrow$ stop time distribution q_ϕ

$$p_\theta(t|\mathbf{y},\mathbf{x}) = \frac{p_\theta(t,\mathbf{y},\mathbf{x})}{p_\theta(\mathbf{y},\mathbf{x})} = \frac{p_\theta(\mathbf{y}|t,\mathbf{x}) \cdot p(t|\mathbf{x}) \cdot p(\mathbf{x})}{p(\mathbf{y},\mathbf{x})}$$

- Likelihood: $p_\theta(\mathbf{y}|t,\mathbf{x})$, and it requires the label $\mathbf{y}$, which is infeasible during the testing phase $\rightarrow$ loss $\ell(\mathbf{y},\mathbf{x}_t;\theta)$

  Minimize: $\mathcal{L}_{\beta-VAE} = E_{z\sim p(z|x)}[-\log q(x|z)] + \beta \dot{K}L(p(z|x)||q(z))$

  $\Rightarrow$

  Minimize:
  $\mathcal{L}_{\beta-VAE}(\theta, q_\phi; \mathbf{x}, \mathbf{y}) := \mathbb{E}_{q_\phi} - \log p_\theta(\mathbf{y}|t,\mathbf{x}) + \beta \mathbf{KL}(q_\phi(t)\|p(t|\mathbf{x}))$

  $\Rightarrow$

  Maximize:
  $\mathcal{J}_{\beta-VAE}(\theta, q_\phi; \mathbf{x}, \mathbf{y}) := \mathbb{E}_{q_\phi} \log p_\theta(\mathbf{y}|t,\mathbf{x}) - \beta \mathbf{KL}(q_\phi(t)\|p(t|\mathbf{x}))$

Variational Bayes Perspective

stop time t	latent variable
label $\boldsymbol{y}$	observation
loss $\ell(\boldsymbol{y}, \boldsymbol{x}_t; \theta)$	likelihood $p_\theta(\boldsymbol{y}\|t, \boldsymbol{x})$
stop time distribution q_ϕ	posterior $p_\theta(t\|\boldsymbol{y}, \boldsymbol{x})$
regularization	prior $p(t\|\boldsymbol{x})$

$\min_{\theta,\phi} \mathcal{L}(\mathcal{F}_\theta, q_\phi; x, y)$ ⟺ (equivalent) $\max_{\theta,\phi} \mathcal{J}_{\beta-VAE}(\mathcal{F}_\theta, q_\phi; x, y)$

(i.e., β-VAE, ELBO)

Figure: Corresponds between proposed model and Bayes' model

Proof of Lemma 1:

$$
\begin{aligned}
\mathcal{J}_{\beta-VAE}(\theta, q_\phi; \mathbf{x}, \mathbf{y}) :&= \mathbb{E}_{q_\phi} \log p_\theta(\mathbf{y}|t, \mathbf{x}) - \beta \mathbf{KL}(q_\phi(t) \| p(t|\mathbf{x})) \\
&= -\mathbb{E}_{q_\phi} \ell(\mathbf{y}, \mathbf{x}_t; \theta) - \beta \mathbb{E}_{q_\phi(t)} \log \frac{q_\phi(t)}{p(t|\mathbf{x})} \\
&= -\mathbb{E}_{q_\phi} \ell(\mathbf{y}, \mathbf{x}_t; \theta) - \beta \mathbb{E}_{q_\phi(t)} \log q_\phi(t) + \beta \mathbb{E}_{q_\phi(t)} \log p(t|\mathbf{x}) \\
&= -[\mathbb{E}_{q_\phi} \ell(\mathbf{y}, \mathbf{x}_t; \theta) - \beta H(q_\phi)] + \beta \mathbb{E}_{q_\phi} \log \frac{1}{T} \\
&= -\mathcal{L}(\theta, q_\phi; \mathbf{x}, \mathbf{y}) - \beta \log T
\end{aligned}
$$

Effective Training Algorithm

VAE method: perform optimization steps over θ (**M** step for learning) and ϕ (**E** step for inference) alternatively until convergence.

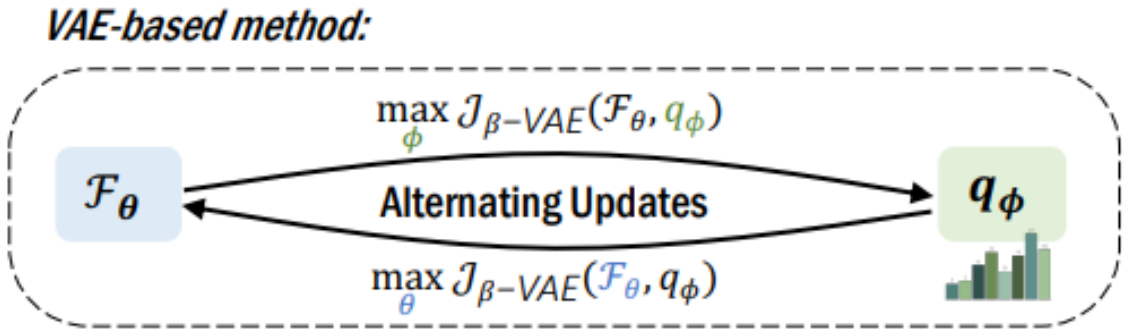


Figure: VAE-based Method

Limitations:

- The alternating training can be slow to converge and requires tuning the training scheduling;
- The inference step for learning q_ϕ may have the mode collapse problem, which in this case means q_ϕ only captures the time step t with highest averaged frequency.

Effective Training Algorithm

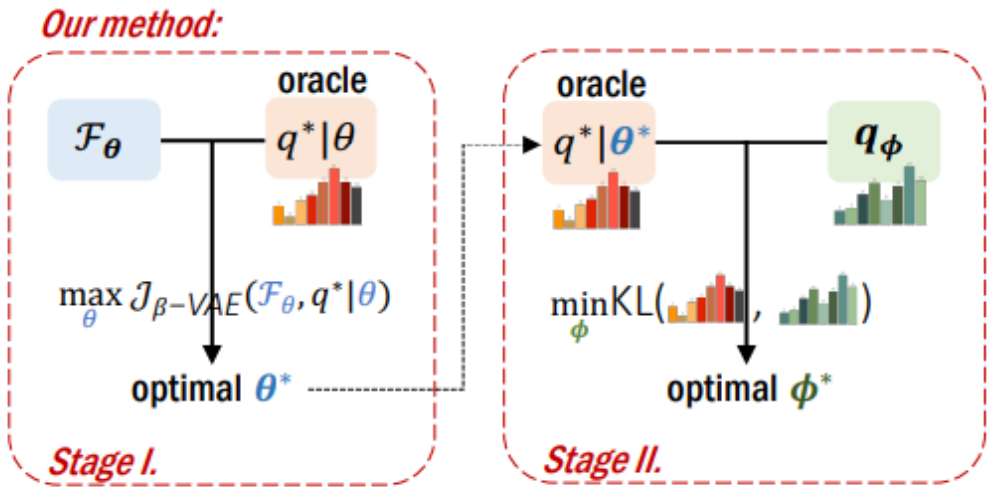


Figure 3. Two-stage training framework.

Stage 1 : Find the optimal θ by maximizing the conditional mariginal likelihood when the stop time distribution follows an oracle distribution q^*_{θ}

Stage 2 : Fix optimal θ, learn the q_{ϕ} to mimic the oracle

Stage 3 :(optimal) Fine-tune θ and ϕ jointly.

Effective Training Algorithm

Oracle Stop Time distribution

For each fixed θ

$$\begin{aligned} q_\theta^*(\cdot|\mathbf{y},\mathbf{x}) &= argmax_{q\in\Delta^{T-1}}\mathcal{J}_{\beta-VAE}(\theta,q_\phi;\mathbf{x},\mathbf{y}) \\ &= argmin_{q\in\Delta^{T-1}}\mathcal{L}(\theta,q_\phi;\mathbf{x},\mathbf{y}) \end{aligned}$$

where $q \in \Delta^{T-1}$ denotes q is a vector of dimension $T-1$;
Then we can get a close form solution:

$$\begin{aligned} q_\theta^*(t|\mathbf{y},\mathbf{x}) &= \frac{p_\theta(\mathbf{y}|t,\mathbf{x})^{\frac{1}{\beta}}}{\sum_{t=1}^{T} p_\theta(\mathbf{y}|t,\mathbf{x})^{\frac{1}{\beta}}} \\ &= \frac{exp(\frac{1}{\beta}\ell(\mathbf{y},\mathbf{x}_t;\theta))}{\sum_{t=1}^{T} exp(\frac{1}{\beta}\ell(\mathbf{y},\mathbf{x}_t;\theta))} \end{aligned} \tag{7,8}$$

Interpretation

- It is the optimal stop time distribution given a predictive model $\mathcal{F}_\theta$
- When$\beta = 1$,the oracle is the true posterior, $q_\theta^*(t|\mathbf{y},\mathbf{x}) = p_\theta(t|\mathbf{y},\mathbf{x})$
- This posterior is computationally tractable, but it requires the knowledge of the true label $\mathbf{y}$
- q_θ^* is a function of $\theta, t, \mathbf{y}, \mathbf{x}$ and independent fromϕ

Effective Training Algorithm

Oracle Stop Time distribution: How to get it?

Minimize:

$$\mathcal{L}(\theta, q_\phi; \mathbf{x}, \mathbf{y}) = \mathbb{E}_{q_\phi}\, \ell(\mathbf{y}, \mathbf{x}_t; \theta) - \beta H(q_\phi)$$
$$= \sum_{t=1}^{T} [-q_\phi(t) \log p_\theta(\mathbf{y}|t, \mathbf{x}) + \beta q_\phi(t) \log q_\phi(t)]$$

we want to get the optimal value of $q_\phi(t)$, while $0 < p_\theta(\mathbf{y}|t, \mathbf{x}) < 1$ is a constant related to t for $q_\phi(t)$. Then the optimize problem is as follows:

$$\textbf{Minimize: } \sum_{t=1}^{T} [-q_\phi(t) \log p_\theta(\mathbf{y}|t, \mathbf{x}) + \beta q_\phi(t) \log q_\phi(t)]$$

where:

$$\sum_{t=1}^{T} q_\phi(t) = 1$$
$$0 \leq q_\phi(t) \leq 1$$

Simply, we use q_t, p_t denote $q_\phi(t)$ and $p_\theta(\mathbf{y}|t, \mathbf{x})$ respectively.
It is an convex optimization problem. Hessian matrix is positive definite matrix and the constraints are linear function. It has a global minimum

Effective Training Algorithm

Oracle Stop Time distribution

Use KKT condition to solve this problem :

$$L(q_t, \lambda, \mu, \sigma) = \sum_{t=1}^{T}[-q_t \log p_t + \beta q_t \log q_t] + \lambda(\sum_{t=1}^{T} q_t - 1) + \sum_{t=1}^{T}(-\mu_t q_t) + \sum_{t=1}^{T} \sigma_t(q_t - 1)$$

KKT conditions are:

$$\nabla_{q_t} L = 0 \tag{4}$$

$$\sum_{t=1}^{T} q_t - 1 = 0 \tag{5}$$

$$-q_t \leq 0, \mu_t \geq 0, \mu_t q_t = 0 \tag{6}$$

$$q_t - 1 \leq 0, \sigma_t \geq 0, \sigma_t(q_t - 1) = 0 \tag{7}$$

For Equation 4:

$$\frac{\alpha L(q_t, \lambda, \mu, \sigma)}{\alpha q_t} = -\log p_t + \beta \log q_t + \beta + \lambda - \mu_t + \sigma_t = 0$$

$$q_t = \exp(\frac{\mu_t - \sigma_t - \beta - \lambda}{\beta}) p_t^{\frac{1}{\beta}}$$

Actually, $\mu_t = 0$ and $\sigma_t = 0$ and use the equation 5, we can get the oracle close form distribution.

$$q_t = \frac{p_t^{\frac{1}{\beta}}}{\sum_{t=1}^{T} p_t^{\frac{1}{\beta}}}$$

Effective Training Algorithm

Stage I. Predictive Model Learning (Learn θ)

Method:

- Normal M step: assume we know when to stop (q_ϕ) and optimize θ, where in the E step we learn q_ϕ assume we know θ
- Proposed model: Use q_θ^* to replace q_ϕ, assuming the stop time always follows the best stopping distribution that depends on θ (only Training time strategy)

Stage I. Oracle model learning

$$\max_\theta \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \mathcal{J}_{\beta-VAE}(\mathcal{F}_\theta, q_\theta^*; x, y) = \max_\theta \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \sum_{t=1}^{T} q_\theta^*(t|y,x) \log \underbrace{p_\theta(y|t,x)}_{\text{likelihood of the output at } t\text{-th layer}}$$

It can be transformed to Expectation form:

$$\max_\theta \frac{1}{\mathcal{D}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathbb{E}_{t\sim q_\theta^*(t|\mathbf{y},\mathbf{x})} \log p_\theta(\mathbf{y}|t,\mathbf{x}) \tag{8}$$

Effective Training Algorithm

Stage II. Imitation with Sequential Policy (Learn ϕ)

Method: Use policy network to mimic the optimal oracle stop time distribution in order to learn π_ϕ

Variational stop time distribution $q_\phi(t|\mathbf{x})$: induced by the sequential policy π_ϕ

Minimizing the **forward KL divergence:** between $q^*_\theta(t|\mathbf{y}, \mathbf{x})$ and $q_\phi(t|\mathbf{x})$

Stage II. Imitation With Sequential Policy

forward KL divergence $$\mathrm{KL}(q^*_{\theta^*}||q_\phi) = -\sum_{t=1}^{T} q^*_{\theta^*}(t|y,x) \log q_\phi(t|x) - H(q^*_{\theta^*})$$

It is the cross-entropy loss since $H(q^*_\theta)$ can be consider as a constant (θ is fixed)

Effective Training Algorithm

Stage II. Imitation with Sequential Policy (Learn ϕ)

Minimizing the **Reverse KL divergence:** between $q^*_\theta(t|\mathbf{y}, \mathbf{x})$ and $q_\phi(t|\mathbf{x})$

$$KL(q_\phi \| q^*_\theta) = \sum_{t=1}^{T} -q_\phi(t) \log q^*_\theta(t|\mathbf{y}, \mathbf{x}) - H(q_\phi)$$

It is equivalent to solving **Maximum-entropy RL**
Reverse KL has mode collapse issue and may lead to a distribution q_ϕ that captures only a common stopping time t for all $\mathbf{x}$ that on average performs the best

Effective Training Algorithm

Stage II. Imitation with Sequential Policy (Learn ϕ)

Mapping

- State: $\mathbf{x}_t$
- Action: $t \sim \pi_t := \pi_\phi(\mathbf{x}, \mathbf{x}_t)$ is a stop/continue decision
- State Transition: it is determined by θ and a_t
- Reward:

$$r(\mathbf{x}_t, a_t, \mathbf{y}) = \begin{cases} -\beta\ell(\mathbf{y}, \mathbf{x}_t; \theta) & if \quad a_t = 0(i.e. stop) \\ 0 & if \quad a_t = 1(i.e. continue) \end{cases}$$

Minimizing $KL(q_\phi \| q_\theta^*)$

$$KL(q_\phi \| q_\theta^*) = \sum_{t=1}^{T} -q_\phi(t) \log q_\theta^*(t|\mathbf{y}, \mathbf{x}) - H(q_\phi)$$

Solving the maximum-entropy RL:

$$\max_\phi \mathbb{E}_{\pi_\phi} \sum_{t=1}^{T} [r(\mathbf{x}_t, a_t, \mathbf{y}) + H(\pi_t)]$$

Effective Training Algorithm

Stage II. Imitation with Sequential Policy (Learn ϕ)

Proof (A.2) (There are some errors):

$$\min_{\phi} \mathbf{KL}(q_\phi(t)||q_\theta^*(t|\mathbf{y},\mathbf{x})) \tag{12}$$

$$= \min_{\phi} -\sum_{t=1}^{T} q_\phi(t) \log q_\theta^*(t|\mathbf{y},\mathbf{x}) - H(q_\phi) \tag{13}$$

$$= \min_{\phi} -\sum_{t=1}^{T} q_\phi(t) \log p_\theta(\mathbf{y}|t,\mathbf{x})^{\frac{1}{\beta}} - H(q_\phi) \tag{14}$$

$$+ \sum_{t=1}^{T} q_\phi(t) \log \sum_{\tau=1}^{T} p_\theta(\mathbf{y}|\tau,\mathbf{x})^{\frac{1}{\beta}} \tag{15}$$

$$= \cdots \tag{9}$$

$$= \max_{\phi} \sum_{t=1}^{T} -q_\phi(t) \frac{1}{\beta} \ell(\mathbf{y},\mathbf{x}_t;\theta) + H(q_\phi) \tag{22}$$

$$= \max_{\phi} \mathbb{E}_{t\sim q_\phi}[-\frac{1}{\beta}\ell(\mathbf{y},\mathbf{x}_t;\theta) - \log q_\phi(t)] \tag{23}$$

Define action: $a_t \sim \pi_t = \pi_\phi(\mathbf{x}, \mathbf{x}_t)$, the reward function as
($\pi(t) \in [0, 1]$ is the probability of stopping and π_ϕ is the parametrized policy.)

$$r(\mathbf{x}_t, a_t; \mathbf{y}) := \begin{cases} -\frac{1}{\beta}\ell(\mathbf{y}, \mathbf{x}_t; \theta), a_t = 1(i.e. stop) \\ 0, a_t = 0(i.e. continue) \end{cases}$$

and Transition probability:

$$P(\mathbf{x}_{t+1}|\mathbf{x}_t, a_t) := \begin{cases} 1, \mathbf{x}_{t+1} = \mathcal{F}_\theta(\mathbf{x}_t) \quad and \quad a_t = 0 \\ 0, else. \end{cases}$$

$$\max_\phi \mathbb{E}_{t\sim q_\phi}[-\frac{1}{\beta}\ell(\mathbf{y}, \mathbf{x}_t; \theta) - \log \underline{q_\phi(t)}] \tag{24}$$

$$= \max_\phi \mathbb{E}_{\pi_\phi} \sum_{t=1}^{T} r(\mathbf{x}_t, a_t; \mathbf{y}) - \log \pi_t(a_t|\mathbf{x}, \mathbf{x}_t) \tag{25}$$

$$= \max_\phi \mathbb{E}_{\pi_\phi} \sum_{t=1}^{T} [r(\mathbf{x}_t, a_t; \mathbf{y}) + H(\pi_t)] \tag{26}$$

Stage IIIThe optimal Fine Tuning Stage

- Comparing with EM algorithm, first two stages can save a lot of alternation steps;
- Then we can fine-tune θ and ϕ jointly towards the $\beta - VAE$ objective.
- Experimentally, it does not improve much the performance.

Implement Details

Let us face the reality: We need to do summation over T layers, leading to a high computation and memory costs

1. **Fewer output channels**: Same as SDN paper, we choose a small number of internal layers instead of output $\mathbf{x}_t$ at all internal layers.
2. **Stochastic sampling in Step I**: Approximate the expectation over q_θ^* : Randomly sample a layer and only compute the $\log p_\theta(\mathbf{y}|t,\mathbf{x})$ at t_s

$$\max_\theta \frac{1}{\mathcal{D}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathbb{E}_{t\sim q_\theta^*} \log p_\theta(\mathbf{y}|t,\mathbf{x}) \tag{10}$$

$\Rightarrow$

$$\max_\theta \frac{1}{\mathcal{D}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \log p_\theta(\mathbf{y}|t_s,\mathbf{x}) \tag{11}$$

When do sampling, $t \sim Uniform(1/T)$ instead of $t \sim q_\theta^*$

Back-propagate through $q_\theta^*(t|\mathbf{y},\mathbf{x})$? theoretically(4.2) but not experimentally (4.5) since we just random sampling.

Implement Details

3 **MAP estimate in Step II**: Approximate MAP estimate instead of distribution q_θ^*

$$\mathbf{KL}(q_\phi \| q_\theta^*) = -\sum_{t=1}^{T} q_\theta^*(t|\mathbf{y}, \mathbf{x}) \log q_\phi(t) - H(q_\theta^*) \tag{10}$$

$\Rightarrow$

$$-\log q_\phi(\hat{t}(\mathbf{x}, \mathbf{y}))$$
$$\textit{where} \quad \hat{t}(\mathbf{x}, \mathbf{y}) = \underset{t \in [T]}{\textit{argmax}}\, q_\theta^*(t|\mathbf{y}, \mathbf{x})$$

Experiments

Experiment tasks:

- Learning to optimize: sparse recovery
- Task-imbalanced meta learning: few-shot learning
- Image Denoising
- Image recognition (NOT SO GOOD)

The question to answer:

- Whether the stopping policy can improve the performance?
- Whether the training algorithm is more effective than EM algorithm?

Experiment I:Sparse Recovery

- Task: Recover $\mathbf{x}^*$ from its noisy measurement $\mathbf{b} = A\mathbf{x}^* + \epsilon$
  - Given: $\mathbf{b} \in \mathbb{R}^m$ is the noisy linear measurement
  - Parameter: $A \in \mathbb{R}^{m \times n}$ and $m << n$
  - $\epsilon \in \mathbb{R}^m$
  - Target: $\hat{y} = \mathbf{x}^* \in \mathbb{R}^n$
- ISTA:
  - LASSO formulation:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1^1$$

Algorithm 1 ISTA

Input: $\mathbf{x}^0$
Output: $\mathbf{x}^L$
  for $l = 0, 1, \ldots, L-1$ **do**
    $\mathbf{x}^{l+1} = S_\lambda((\mathbf{I} - \frac{1}{\mu}\mathbf{A}^T\mathbf{A})\mathbf{x}^l + \frac{1}{\mu}\mathbf{A}^T\mathbf{b})$
  end for

Let: $\mathbf{A}_t^2 = \mathbf{I} - \frac{1}{\mu}\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}_t^1 = \frac{1}{\mu}\mathbf{A}^T$

$$\mathbf{x}^{l+1} = S_\lambda(\mathbf{A}_t^2\mathbf{x}^l + \mathbf{A}_t^1\mathbf{b})$$

$$S_\lambda(x) = sign(x) \cdot max\{|x| - \lambda, 0\}$$

Experiment I:Sparse Recovery

- ✓ LISTA: a T-layer network with update steps (t is the layer):

$$\mathbf{x}_t = \eta_{\lambda_t}(W_t^1 \mathbf{b} + W_t^2 \mathbf{x}_{t-1}), \quad t = 1, \cdots, T, \tag{11}$$

- ✓ Rhe predictive sparse for the $t - th$ layer: $\hat{y}_t = x_t$;
- ✓ Loss is:

$$\ell(\mathbf{y}, \mathbf{x}_t; \theta) = \frac{1}{N} \sum_{n=1}^{N} \|\hat{\mathbf{y}}_t^n(b^n, \mathbf{x}_t^n; \theta) - \mathbf{y}^n\|_2^2 \tag{12}$$

- ✓ The recovery performance is evaluated by NMSE (in dB), smaller is better:

$$NMSE(\hat{x}, x^*) = 10 \log_{10}\left(\frac{\mathbb{E}\|\hat{x} - x^*\|_2^2}{\mathbb{E}\|x^*\|_2^2}\right)$$

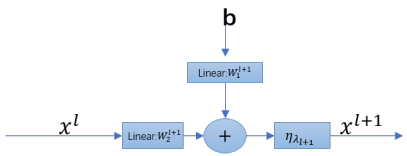


Figure: A Single Network Layer in LISTA

Learnable parameters in the original network:

$$\theta = \{(\lambda_t, W_t^1, W_t^2)\}_{t=1}^{t=T}$$

Experiment I:Sparse Recovery

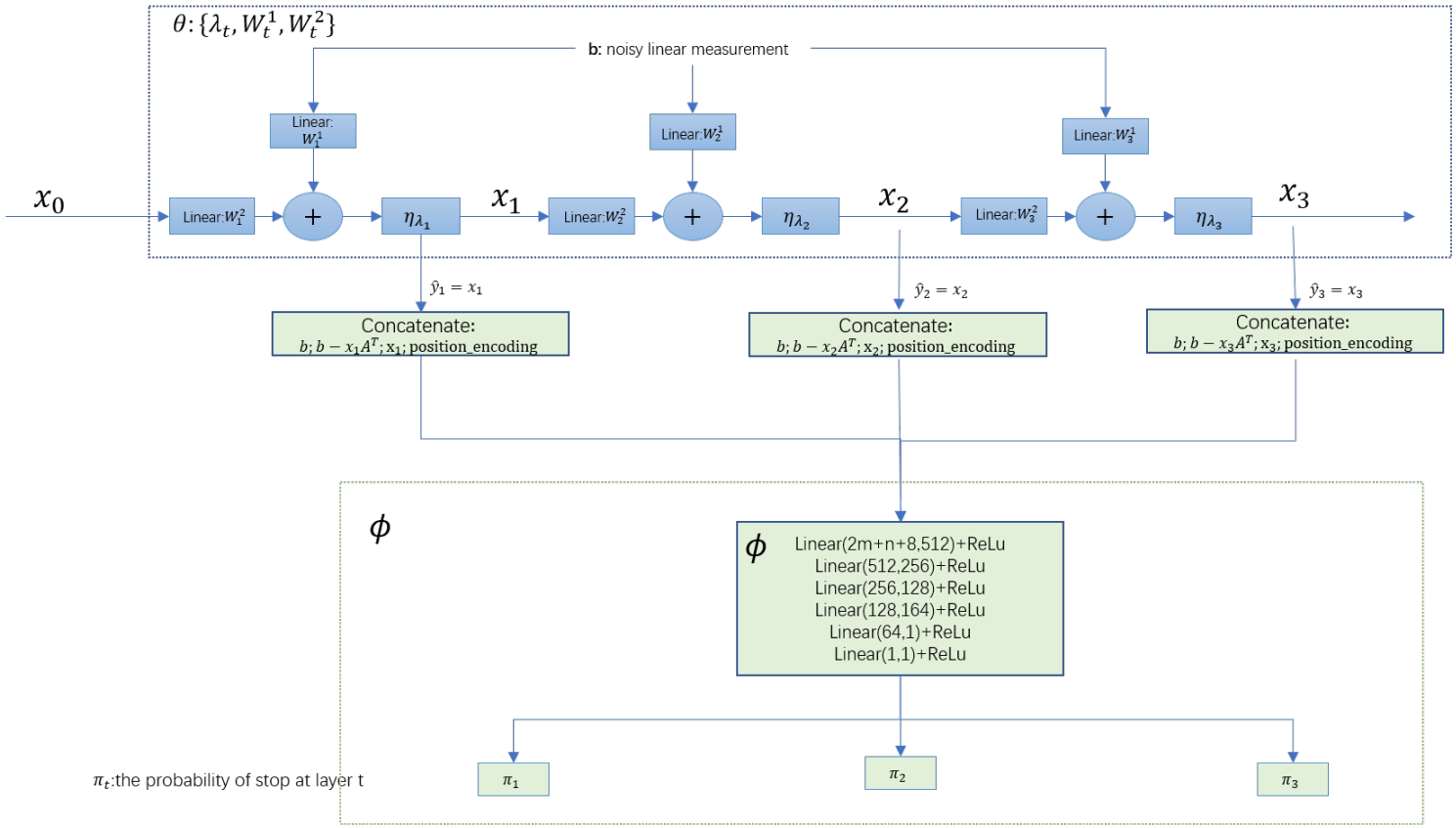


Figure: LISTA-stop Model

Experiment I:Sparse Recovery

Recovery Performance:

1. Leaning based model (LISTA) has a better performance, especially for more difficult tasks;
2. Improvements: LISTA-stop (proposed model) significantly improve the recovery performance
3. Better convergence: LISTA-stop with ≤ 20 iterations performs better than ISTA and FISTA with 100 iterations, which indicates a better convergence.

- SNR: signal-to-noise ration for each sample, which is smaller and the task is more difficult;
- Evaluation Metric: NMSE(in dB), smaller is better;

Table 2. Recovery performances of different algorithms/models.

SNR	mixed	20	30	40
FISTA ($T = 100$)	-18.96	-16.75	-20.46	-20.97
ISTA ($T = 100$)	-14.66	-13.99	-14.99	-15.07
ISTA ($T = 20$)	-9.17	-9.12	-9.24	-9.16
FISTA ($T = 20$)	-11.12	-10.98	-11.19	-11.19
LISTA ($T = 20$)	-17.53	-16.53	-18.07	-18.20
LISTA-stop ($T \leqslant 20$)	**-22.41**	**-20.29**	**-23.90**	**-24.21**

Experiment I:Sparse Recovery

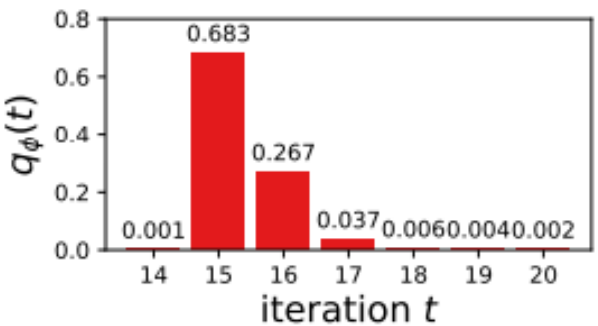
Stopping distribution $q_\phi(t)$: (a) shows an average stopping distribution over the test samples.

$$\frac{1}{\mathcal{D}_{test}} \sum_{\mathbf{x} \in D_{test}} q_\phi(t|\mathbf{x})$$

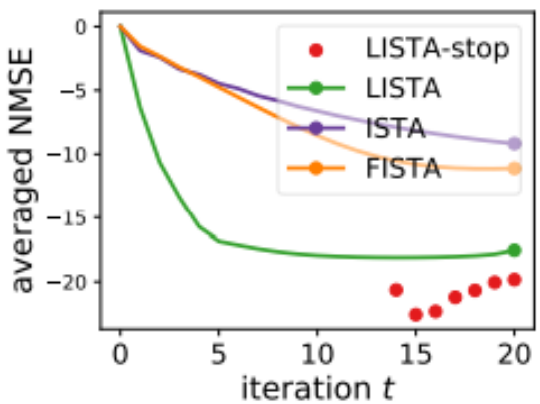
Finding: With a high probability LISTA-stop terminates the process before arriving at 20-th iteration

Convergence comparison: (b) shows the change of NMSE as the number of iterations increases. Red dots: the NMSE weighted by the stopping distribution q_ϕ (In the first 13 iterations $q_\phi = 0$)
Finding:

1. Later stopped problems are more difficult to solve: expected NMSE increases as the number of parameters increase.
2. 15th-generation, NMSE is the lowest while the average stop probability is the highest.



(a) stop time distribution



(b) convergence

Experiment I:Sparse Recovery

Ablation study on training algorithms: Comparing with AEVB algorithm, which jointly optimize $\mathcal{F}_\theta$ and q_ϕ.
Findings:

- q_ϕ in AEVB gradually becomes concentrated on one layer and does not fet rid of the local minimum
- Stage III does not improve much of the performance of the two-stage training

Table 3. Different algorithms for training LISTA-stop.

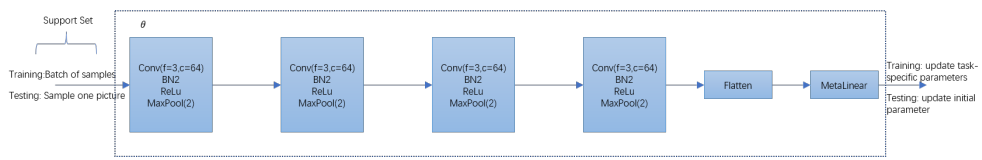
SNR	mixed	20	30	40
AEVB algorithm	-21.92	-19.92	-23.27	-23.58
Stage I. + II.	**-22.41**	**-20.29**	**-23.90**	**-24.21**
Stage I.+II.+III.	**-22.78**	**-20.59**	**-24.29**	**-24.73**

Experiment II:Task-imbalanced Meta Learning

- ✓ Task: Task-imbalance few-shot Learning. Each task contains k-shots for each class where k can vary k_1-k_2-shot
- ✓ MAML-stop:
  - Built on top of MAML, find the best number of iteration (gradient update steps) in the meta-learning setting,especially for the task-imbalanced meta learning
  - Intuitively, the tasks with less training data would prefer fewer steps of gradient-update to prevent over fitting;
- ✓ Dataset:
  - Omniglot:20 instances of 1623 characters from 50 different alphabets
  - MiniImagenet: 64 training classes,12 validation classes, and 24 test classes (imbalance issue is more severe)

Experiment II:Task-imbalanced Meta Learning

- MAML: one or few gradient adaptation steps during the training. In practice, it uses 5 and 10 steps during meta-training and meta-testing stages for the miniImageNet experiment.
- MAML-stop: The total number of gradient adaptation steps T= 14, and we find the optimal gradient adaptation step among these 14 steps.
- Iter: the number of gradient adaptation steps (There are T steps in the figure)
- Backbone: a CNN model with 4 convolutional layers,the same color denotes the same parameters at that time



Experiment II:Task-imbalanced Meta Learning

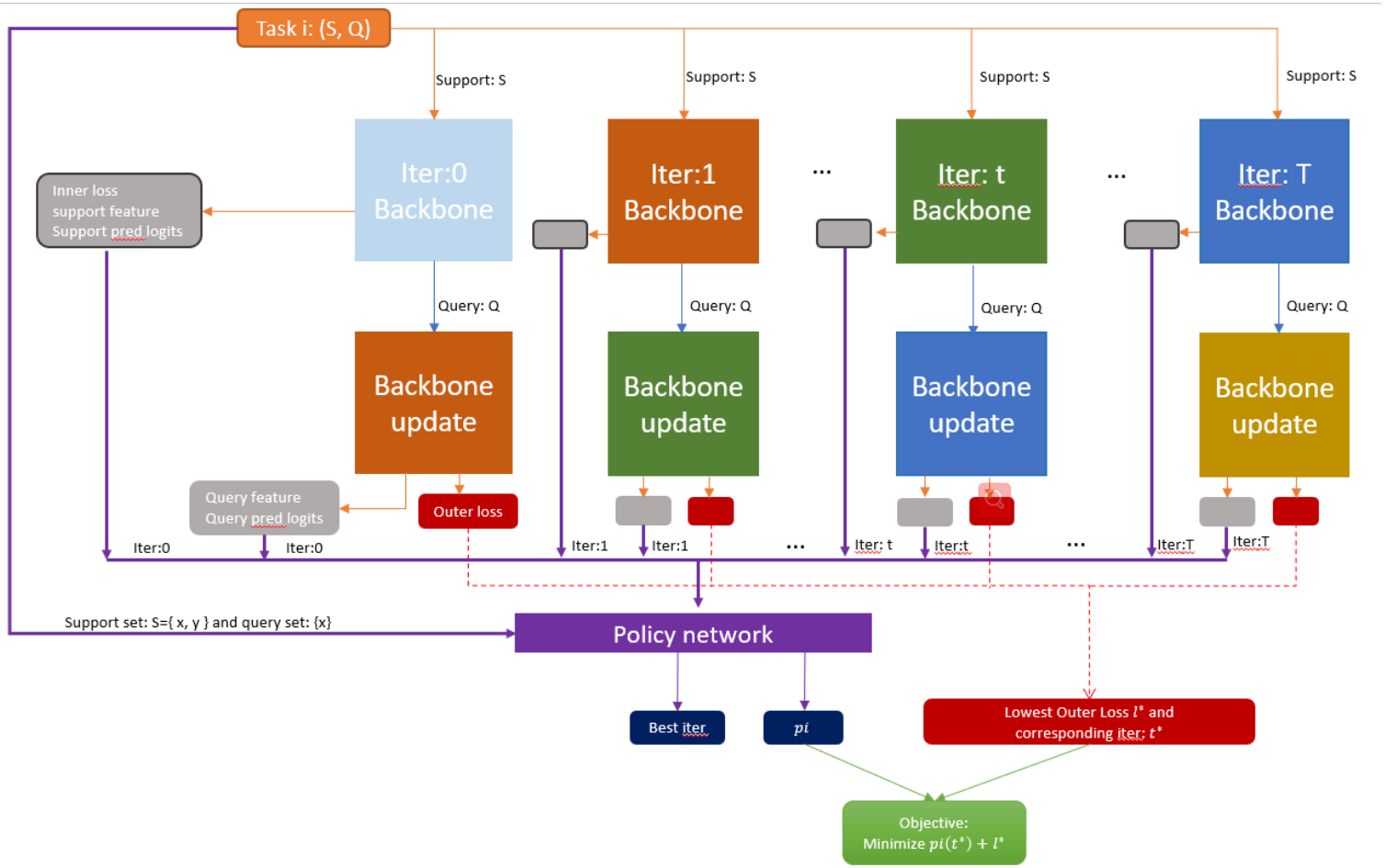


Figure: MAML-stop Model

Experiment II:Task-imbalanced Meta Learning

Input of policy network. The input of the policy network would be:

- $\{(\ell_{tr}(t), Feat_{tr}(t), Pred_{tr}(t))|_{t=0}^{t=T}\}$
- $\{(Feat_{ts}(t), Pred_{ts}(t))|_{t=0}^{t=T}\}$
- the support set $\{X_s, y_s\}$ (images, and labels) and query samples X_q (only images, without label information)

Output of policy network. There are two parts in the output of policy network:

- The best iter, t_{pred}, namely the best number of gradient adaptation steps.
- The posterior distribution of stop time $pi(t)$.

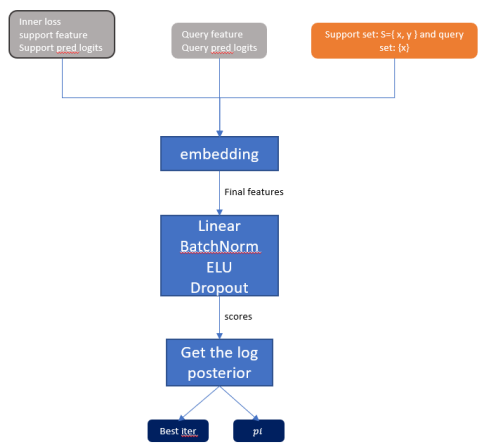


Figure: MAML-stop-policy Model

Experiment II:Task-imbalanced Meta Learning

The specifications are as follows:

- t: # of gradient decent steps, $t^* = \ell_{ts}(t)$;
- x_t: parameters in time step t;
- y: the ground truth of class label. $\hat{y}_t = x_t$: the predictive label of the model updated by the $t - th$ gradient decent step;
- θ: meta-parameters in MAML, specifically those weights in backbone;
- ϕ: parameters in the policy network;
- $p_\theta(y|t, x)$: the probability of predicted results generating from the model whose parameters has t's updated is equal to the ground truth
- $\ell(y, x_t; \theta)$: the experiment is an image recognition task and the final loss is entropy loss. The final target is to update initialize parameter using loss $\ell_{ts}(t; \theta)$, where

$$\ell_{ts}(t; \theta) = \ell(\mathcal{A}lg_t(X_s; \theta), X_q) \tag{13}$$

where ℓ is the cross entropy. $\mathcal{A}lg_t(D_s; \theta)$ generates the updated model parameters (initialized as θ) after t steps according to support set $D_s = \{X_s, y_s\}$. Similarly, D_q is the query set.

Experiment II:Task-imbalanced Meta Learning

Results:

- ✓ Table 4 summarizes the accuracy and 95% confidence interval on the held-out tasks.
  - MAML-stop outperforms than vanilla MAML consistently
  - The accuracy improvement of MiniImagenet is 3.5%
- ✓ Table 5 is a complete result setting where all tasks have the same number of observations
  - MAML-stop still achieves comparable or better performance

Task-imbalanced setting:

Table 4. Task-imbalanced few-shot image classification.

	Omniglot 20-way, 1-5 shot	MiniImagenet 5-way, 1-10 shot
MAML	$97.96 \pm 0.3\%$	$57.20 \pm 1.1\%$
MAML-stop	$\mathbf{98.45 \pm 0.2\%}$	$\mathbf{60.67 \pm 1.0\%}$

Vanilla setting:

Table 5. Few-shot classification in vanilla meta learning setting (Finn et al., 2017) where all tasks have the same number of data points.

	Omniglot 5-way		Omniglot 20-way		MiniImagenet 5-way	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MAML	$98.7 \pm 0.4\%$	$99.1 \pm 0.1\%$	$95.8 \pm 0.3\%$	$98.9 \pm 0.2\%$	$48.70 \pm 1.84\%$	$63.11 \pm 0.92\%$
MAML-stop	$\mathbf{99.62 \pm 0.22\%}$	$\mathbf{99.68 \pm 0.12\%}$	$\mathbf{96.05 \pm 0.35\%}$	$\mathbf{98.94 \pm 0.10\ \%}$	$\mathbf{49.56 \pm 0.82\%}$	$\mathbf{63.41 \pm 0.80\%}$

Experiment III:Image Denoising

- ✓ Task: Task: Recover a clean image $\mathbf{y}$ from a noisy observation $\mathbf{x}$..

$$\mathbf{x} = \mathbf{y} + \mathbf{v} \tag{14}$$

- ✓ Assumption: $\mathbf{v}$ is additive white Gaussian noise (AWGN) with standard deviation σ

Figure: Ground Truth Image

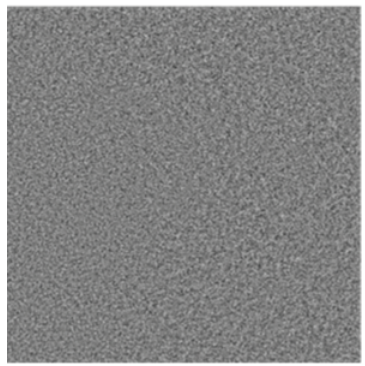


Figure: Noise Signal

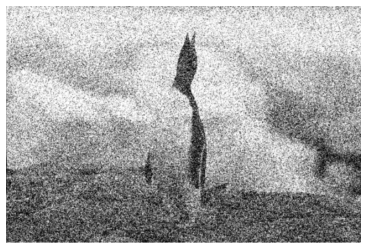


Figure: Noisy Image

Experiment III:Image Denoising

- DnCNN:
  - Based on VGG, using residual learning and batch normalization layer
  - The residual mapping is

$$\mathcal{R}(\mathbf{x}) \approx \mathbf{v} \tag{15}$$

    and

$$\mathbf{y} = \mathbf{x} - \mathcal{R}(\mathbf{x}) \tag{16}$$

  - Then the loss is

$$\ell(\theta) = \frac{1}{2N} \sum_{i=1}^{N} \|\mathcal{R}(\mathbf{x}_i; \theta) - (\mathbf{x}_i - \mathbf{y}_i)\|_2^2 \tag{17}$$

    with $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ represents N noisy-clean training pairs.
- baseline:
  - DL models: DnCNN [8] and $UNLNET_5$; blind Gaussian denoising setting: the noise-level is not given to the model
  - Traditional methods: WNNM, BM3D; need noise-level

Experiment III:Image Denoising

The detailed architecture is as follows:

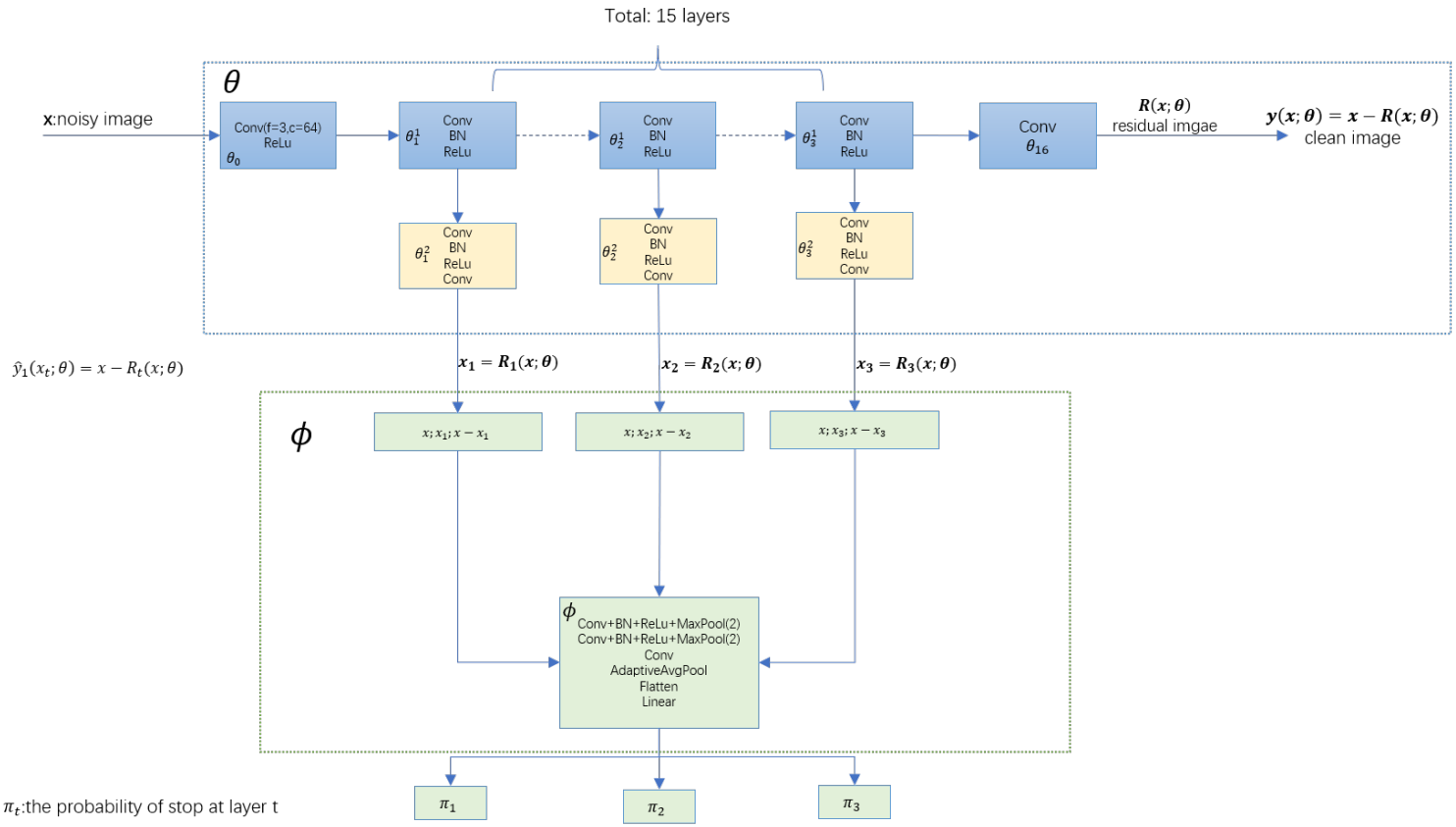


Figure: DnCNN-stop Model

Experiment III:Image Denoising

✓ Results:

- DnCNNstop performs better than the original DnCNN;
- Better generalization ability: for images with noise levels 65 and 75 which are unseen during training phase, DnCNN-stop generalizes significantly better than DnCNN alone

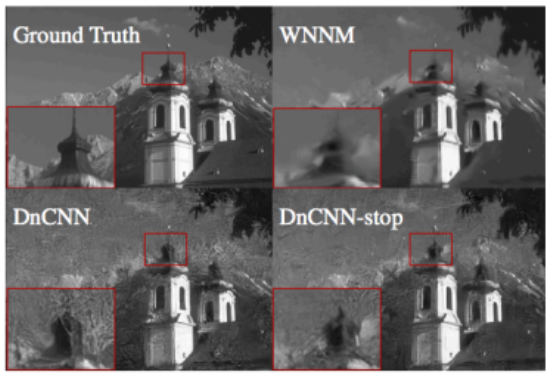


Figure 5. Denoising results of an image with noise level 65. (See Appendix B.3.2 for more visualization results.)

**Noise-level 65, 75 are not observed during training.*

σ	DnCNN-stop	DnCNN	UNLNet$_5$	BM3D	WNNM
35	**27.61**	27.60	27.50	26.81	27.36
45	**26.59**	26.56	26.48	25.97	26.31
55	**25.79**	25.71	25.64	25.21	25.50
*65	**23.56**	22.19	-	24.60	**24.92**
*75	**18.62**	17.90	-	24.08	**24.39**

Experiment IV:Image Recognition

- Dataset: The Tiny ImageNet dataset consists of a subset of ImageNet images, re-sized at 64x64 pixels and RGB. There are 200 classes
- Model: VGG-16
- Different architectures of policy network used in the image recognition inI2stop paper although the final results are not better than SDN results.
    1. MulticlassNetImage
    2. MNIconfidence
    3. imiconfidence

Experiment IV:Image Recognition

Table: VGG-16 Model under the TinyImage Dataset

# layer	VGG-16	"Conv" Output dimension	IC
	input (64 ×64 × 3)		0/1
1	conv3-64	64×64 × 64	1
2	conv3-64	64×64 × 64	1
	maxpool	32×32 × 64	
3	conv3-128	32×32 × 128	1
4	conv3-128	32×32 × 128	1
	maxpool	16×16 × 128	
5	conv3-256	16×16 × 256	1
6	conv3-256	16×16 × 256	1
7	conv3-256	16 ×16 × 258	1
	maxpool	8×8 × 256	
8	conv3-512	8×8 × 512	1
9	conv3-512	8×8 × 512	1
10	conv3-512	8×8 × 512	1
	maxpool	4×4 × 512	
11	conv3-512	4×4 × 512	1
12	conv3-512	4×4 × 512	1
13	conv3-512	4×4 × 512	1
	maxpool	2×2 × 512	
	flatten	2048	
14	FC-4096	2048	1
15	FC-4096	2048	
16	FC-1000	200	
	softmax	200	

Experiment IV:Image Recognition

The detailed architecture is as follows (we take MulticlassNetImage as an example):

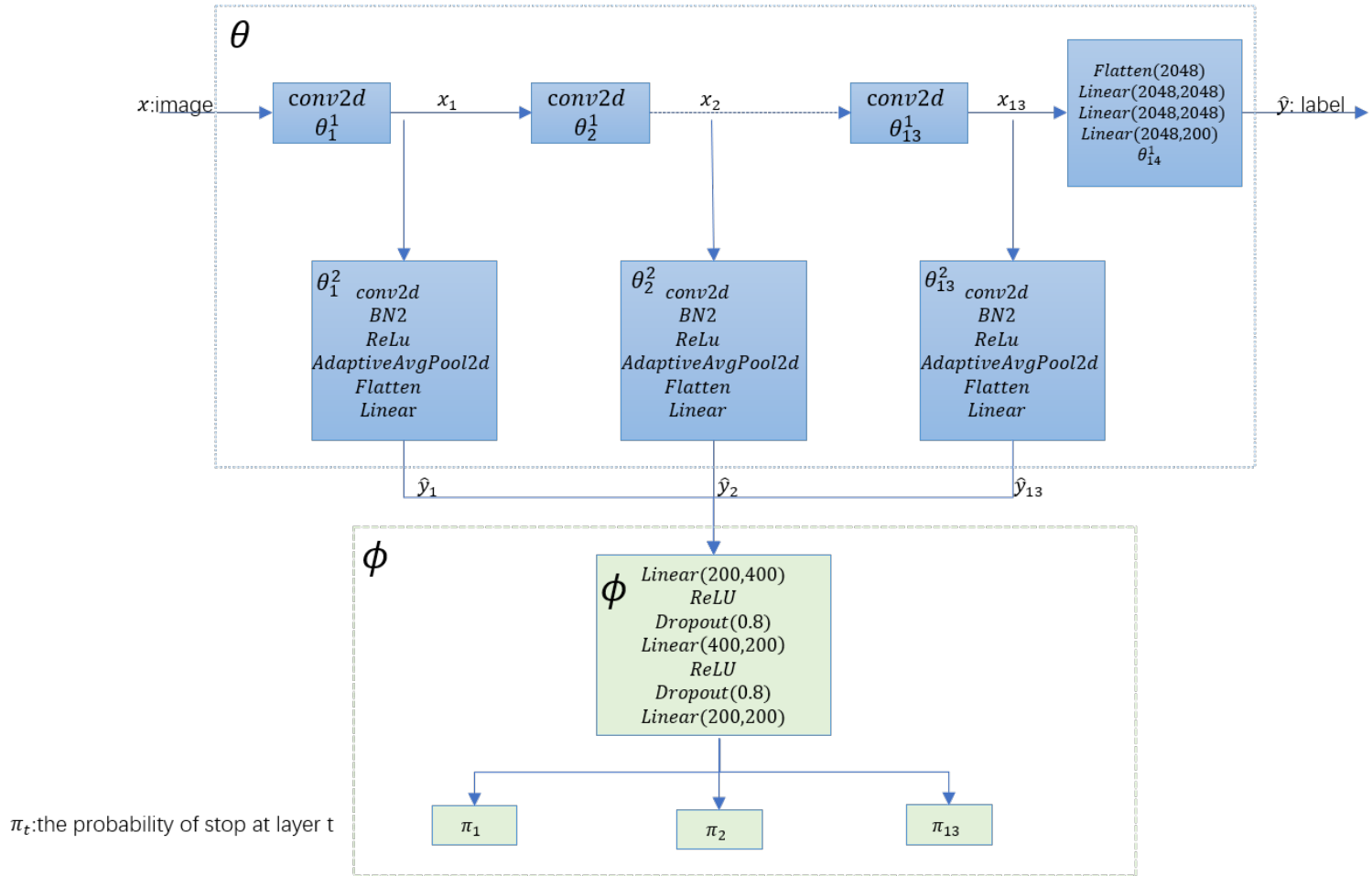


Figure: VGG-stop Model

Experiment IV:Image Recognition

✓ Result:

- The best layer's output accuracy performs better than original VGG (Stage I)
  - Use oracle q_θ to determine the stop time t. The accuracy improved to 83.26%
  - SDN uses confidence to determine the stop time t. The accuracy improved from original VGG to 77.8%
  - Comparison: SDN-Training Loss needs hand-tuning weight τ_i while proposed model directly use the oracle. Also, proposed model has a higher performance.

$$\min_{\theta_{IC}} \mathcal{L} = \min_{\theta_{IC}} \sum_{i=1}^{7} \tau_i \mathcal{L}_i$$

- It is very hard to mimic oracle q_θ^* by π_ϕ (Stage II)
  - Learned π_ϕ is similar accuracy as the heuristic policy in SDN.

Table 7. Image recognition with oracle stop distribution.

VGG16	SDN training	Our Stage I. training
58.60%	77.78% (best layer)	83.26% (best layer)

Conclusion

1. Propose a learning to stop method with a generic framework, which can be applied to diverse range of applications
2. A **Variational Bayes** perspective gives readers a better understanding for the proposed model and joint training.
3. A **Reinforcement Learning** perspective gives a good understanding for the stopping policy (although this paper is not the first)
4. A principled algorithm for jointly leaning the predictive model and the stopping policy is proved efficient
5. Sufficient experiments demonstrate the effectiveness of the proposed model in terms of both the **prediction accuracy** and **inference efficiency**.

Reference I

Sebastian Becker, Patrick Cheridito, and Arnulf Jentzen.

Deep optimal stopping.

Journal of Machine Learning Research, 20:74, 2019.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio.

Generating sentences from a continuous space.

arXiv preprint arXiv:1511.06349, 2015.

Chelsea Finn, Pieter Abbeel, and Sergey Levine.

Model-agnostic meta-learning for fast adaptation of deep networks.

In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

Katerina Fragkiadaki.

Maximum entropy reinforcement learning.

https://www.andrew.cmu.edu/course/10-403/slides/S19maxentRL.pdf.

Dibya Ghosh.

An introduction to control as inference.

https://dibyaghosh.com/blog/rl/controlasinference.html.

Vishal Monga, Yuelong Li, and Yonina C Eldar.

Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing.

arXiv preprint arXiv:1912.10557, 2019.

Reference II

Hung yi Lee.

Meta learning(part 1.

http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2019/Lecture/Meta1%20(v6).pdf.

Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang.

Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising.

IEEE transactions on image processing, 26(7):3142–3155, 2017.