Deep Sequential Neural Networks

Extending Neural Networks with Reinforcement Learning

Ludovic Denoyer, Patrick Gallinari
University Pierre et Marie Curie, Paris, France

Motivations

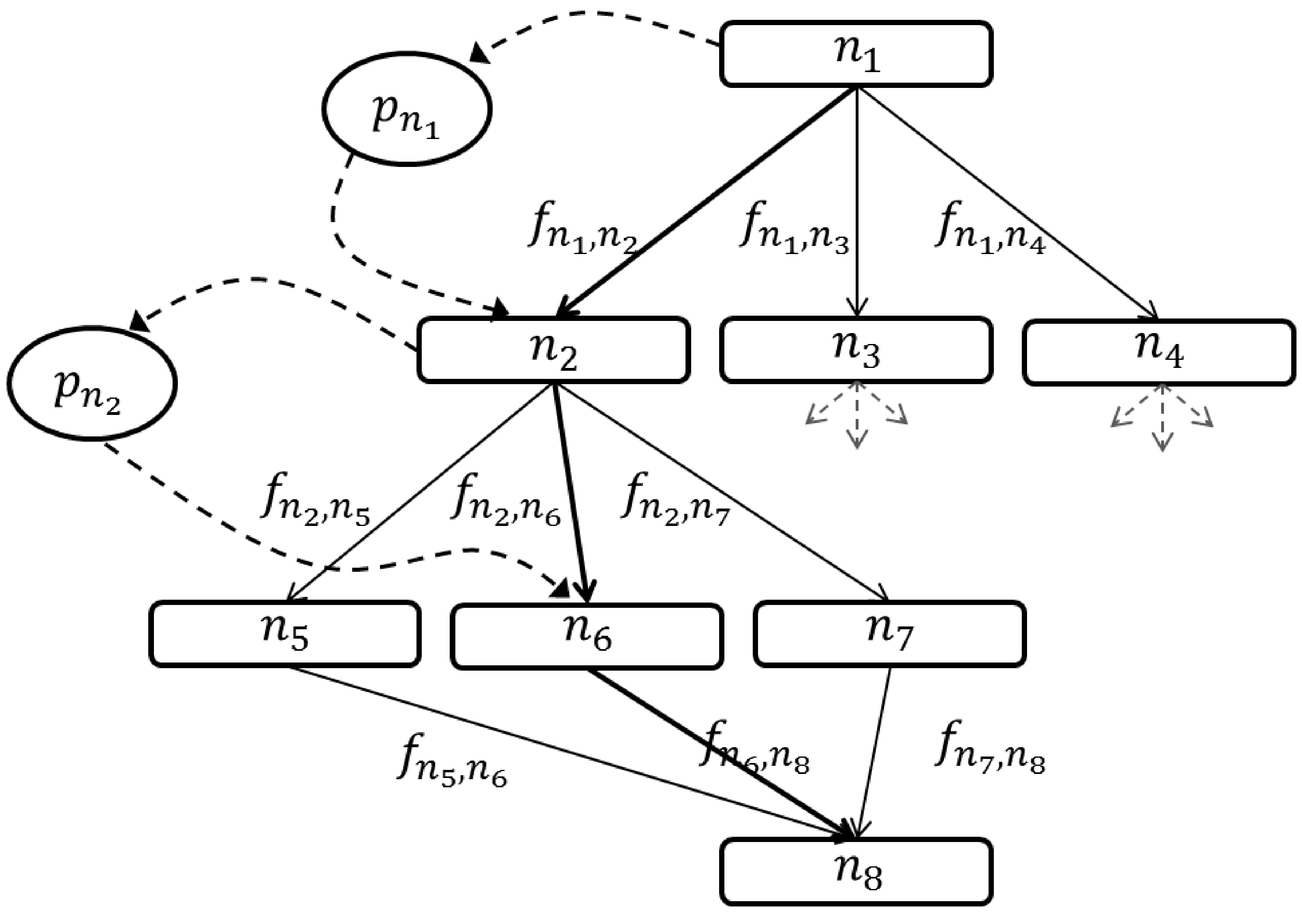
Deep Neural Networks are **sequential processing machines** so:

1. Can we learn Deep Neural Networks using **Reinforcement Learning algorithms** ?
2. Can we extend Deep Neural Networks by considering **multiple processing actions** at each step ?

Contributions

- An **extension of Deep Neural Networks** where the process can choose, at each step, how to process the input datum
  - Two different inputs will not be processed in the same manner
  - In its simplest shape, a Deep Sequential Neural Network is equivalent to a Neural Network
- A learning algorithm inspired by **policy-gradient** that extends classical Neural Networks gradient techniques.

Architecture (DAG)



Inference Algorithm

1. At first, an input $x \in \mathcal{X}$ is presented at the root node $n^{(1)} = n_1$ of the DAG
2. Then, based on x, a child node $n^{(2)}$ is sampled using the $P(c_{(1)_{,.}}|x)$ distribution computed through the p_{n_1} function.
3. The model computes a new representation at node $n^{(2)}$ using $f_{n^{(1)},n^{(2)}}(x)$. A child node of $n^{(2)}$ is sampled following $P(c_{(2)_{,.}}|x)$, ...
4. The same process is repeated until a leaf node. The vector computed at the leaf node level is the output of the model.

Inference Algorithm

procedure Inference(x) ▷ x is the input vector
  $z^{(1)} \leftarrow x$
  $n^{(1)} \leftarrow n_1$
  $t \leftarrow 1$ **while** *not leaf*$(n^{(t)})$ **do**
    Inference finished
    $a^{(t)} \sim p_{n^{(t)}}(z^{(t)})$
    $n^{(t+1)} \leftarrow c_{n^{(t)},a^{(t)}}$
    $z^{(t+1)} \leftarrow f_{n^{(t)},n^{(t+1)}}(z^{(t)})$
    $t \leftarrow t+1$

  return $z^{(t)}$
end procedure

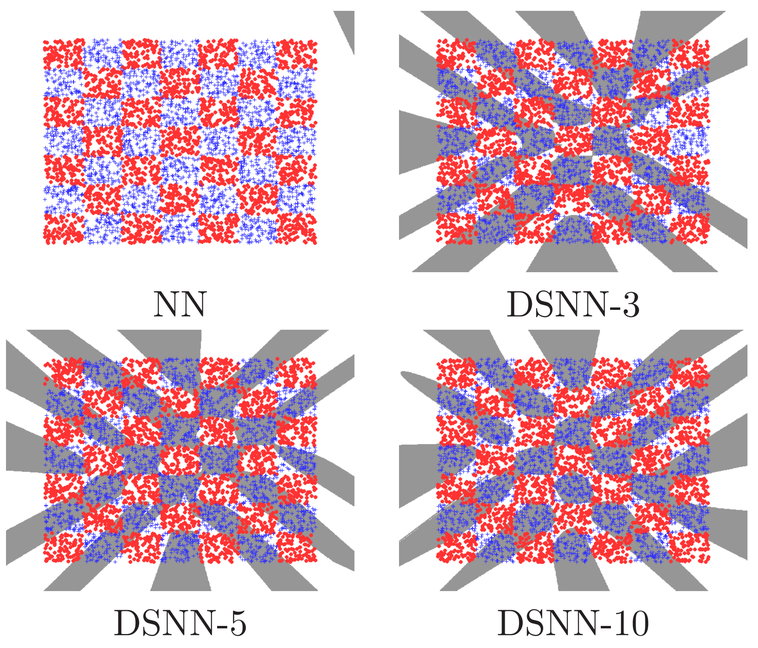
Learning

$$J(\theta,\gamma) = E_{P(x,H,y)}[\Delta(F(x,H),y)] \tag{1}$$

$$\begin{aligned}\nabla_{\theta,\gamma} J(\theta,\gamma) = &\int P(H|x)\nabla_{\theta,\gamma}\left(logP(H|x)\right)\Delta(F(x,H),y)P(x,y)dHdxdy \\ &+ \int P(H|x)\nabla_{\theta,\gamma}\Delta(F(x,H),y)P(x,y)dHdxdy\end{aligned} \tag{2}$$

$$\nabla_{\theta,\gamma} J(\theta,\gamma) \approx \frac{1}{\ell}\sum_{i=1}^{\ell}\left[\frac{1}{M}\sum_{k=1}^{M}\nabla_{\theta,\gamma}\left(logP(H|x_i)\right)\Delta(F(x_i,H),y) + \nabla_{\theta,\gamma}\Delta(F(x_i,H),y)\right] \tag{3}$$

CheckerBoard

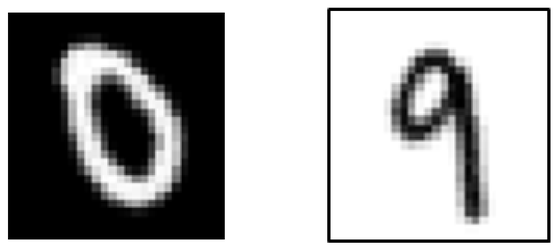


MNIST

	NN	DSNN-2	DSNN-3
nhl	89.4	89.4	89.4
25	93.7	93.6	94.2
25-25	93.6	93.4	93.5
100	95.3	95.4	95.3
100-100	94.6	94.6	94.7

N.B: Digits have been rescaled.

MNIST (Variation)



What happens if input data are coming from different distributions?

	NN	DSNN-2	DSNN-3
nhl	27.7	88.3	88.2
5	37.4	82.6	**83.5**
10	83.4	**89.2**	85.6
10-10	81.1	**85.3**	84.0
25	**91.9**	91.5	91.0
25-25	**90.9**	90.4	85.1
50-50	92.8	**93.5**	92.9

N.B: Colors of half of the digits have been inverted.

Conclusion &Perspectives

- DSNN is **an extension of neural networks** able to choose which transformation to apply, at each timestep, to an input datum.
- DSNN are learned by a **policy gradient**-inspired algorithm.
- Preliminary results over classical datasets are promising.

Perspectives: What happens if each branch of the tree corresponds to a particular NN architecture ? Is the model able to choose the best architecture for any input ?