EURECOM
Sophia Antipolis

PERFUME erc
European Research Council



Learning to Team Play

One World SP Seminars
Aug. 24th, 2020

David Gesbert
EURECOM – Sophia Antipolis, France

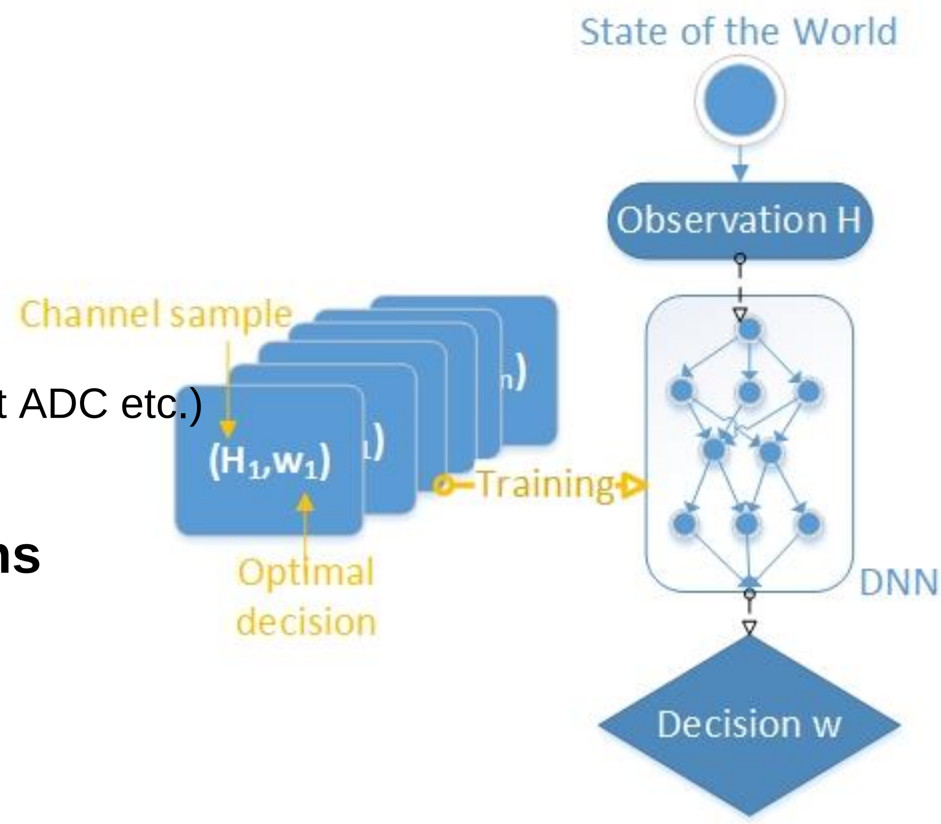
Joint work with Matteo Zecchin, Paul de Kerret

AI for Wireless

AI for Wireless

1. **Go where conventional math models can't go**
2. **Leveraging large measurement data sets**
3. **Leveraging GPUs**
4. **Dealing with <u>uncertainties</u>**

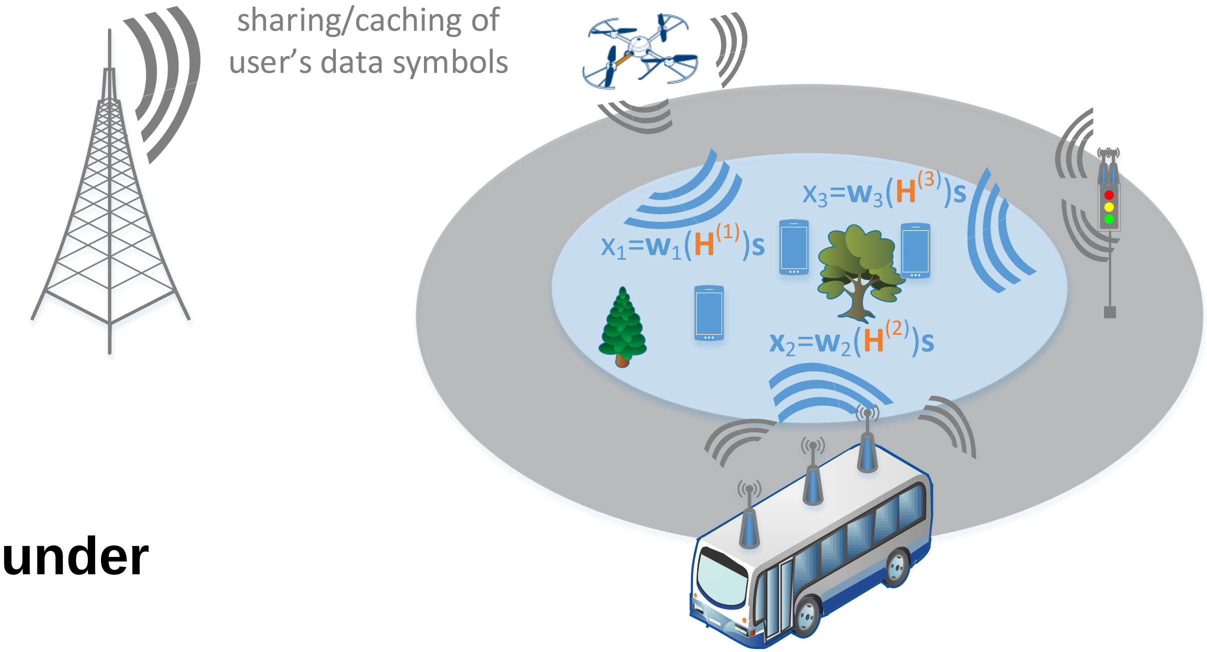
- **PHY applications**
  - Auto-encoding
  - H/W impairment compensation (1 bit ADC etc.)
  - Modulation detection
- **LINK/Network level applications**
  - Fault detection
  - (Predictive) resource allocation
  - SDN optimization
  - <u>Decentralized Edge Cooperation</u>



EURECOM Sophia Antipolis

Decentralized Edge Cooperation

- **Edge Cooperation**
  - Pilot allocation
  - Interference management
  - beam alignment
  - resource allocation
  - Caching
  - Robotic cooperation
    - Self-driving cars
    - Factory robots



- **Decentralized decision under uncertainties**
  - Local observations are noisy, exchanged information are quickly outdated
  - **Need to predict decisions of other devices – but other's decisions also based on noisy predictions ☹**

AI territory!

Building up intuition for team decision: The car crossing example

High-end car (high quality sensors)

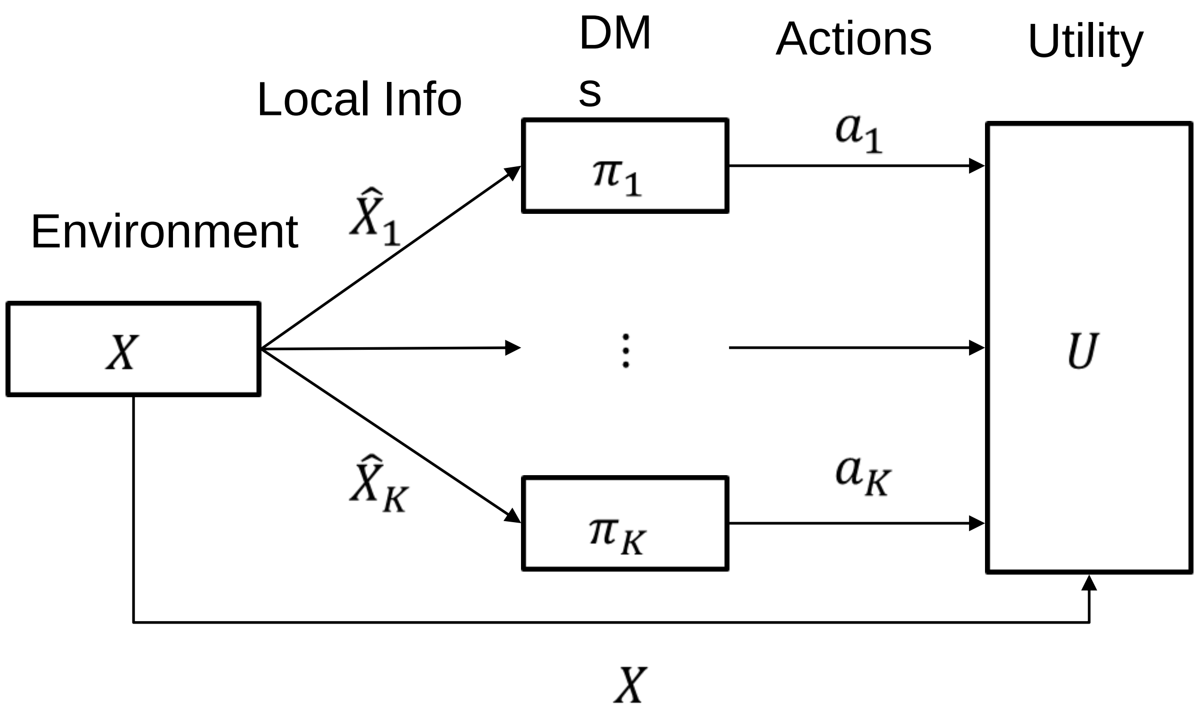
Low-end car (low quality sensors)

Problem: Optimize brake/acceleration policy at each car

Maximize traffic flow under given crash probability threshold

Account for sensor uncertainties

Formalizing the team decision problem



- $X \sim P_X$
- $\hat{X}_1, \dots, \hat{X}_K \sim P_{\hat{X}_1,\dots,\hat{X}_K|X}$

- $\pi_i : \hat{X}_j \rightarrow a_j$
- $U : X \times \prod_{i=1}^{K} a_i \rightarrow \mathbb{R}$

Team decision problem: the goal

For a given distribution $P_{X,\hat{X}_1,\ldots,\hat{X}_K}$ and a set of policies $\{\pi_i\}_{i=1}^{K}$, the average utility is

$$\mathrm{E}_{P_{X,\hat{X}_1,\ldots,\hat{X}_K}}[U(x,\pi_1(\hat{x}_1),\ldots,\pi_K(\hat{x}_K))]$$

-

As system designers, our goal is to find

$$(\pi_1^*,\ldots,\pi_K^*) = \underset{\pi_1,\ldots,\pi_K}{\mathrm{argmax}}\ \mathrm{E}_{P_{X,\hat{X}_1,\ldots,\hat{X}_K}}[U(x,\pi_1(\hat{x}_1),\ldots,\pi_K(\hat{x}_K))]$$

Conventional strategies

- **Benchmarck strategy 1: Naïve (non robust)**

At agent1:

$$(\boldsymbol{\pi}_{\mathbf{1}}^*, \dots, \pi_K^*) = \underset{\pi_1,\dots,\pi_K}{\operatorname{argmax}} \left[U(\hat{x}_1, \pi_1(\hat{x}_1), \dots, \pi_K(\hat{x}_1))\right]$$

- **Benchmarck strategy 2: Naïve (robust to local noise)**

At agent 1:

$$(\boldsymbol{\pi}_{\mathbf{1}}^*, \dots, \pi_K^*) = \underset{\pi_1,\dots,\pi_K}{\operatorname{argmax}} \mathrm{E}_{P_{X,\hat{X}_1,\dots,\hat{X}_K}}\left[U(x, \pi_1(\hat{x}_1), \dots, \pi_K(\hat{x}_1)\right)$$

Team Deep Neural Networks

The solution to the TD problem entails the difficult optimization problem

$$(\pi_1^*, \dots, \pi_K^*) = \underset{\pi_1,\dots,\pi_K}{\operatorname{argmax}} \; \mathrm{E}_{P_{X,\hat{X}_1,\dots,\hat{X}_K}}[U(x, \pi_1(\hat{x}_1), \dots, \pi_K(\hat{x}_K))]$$

-

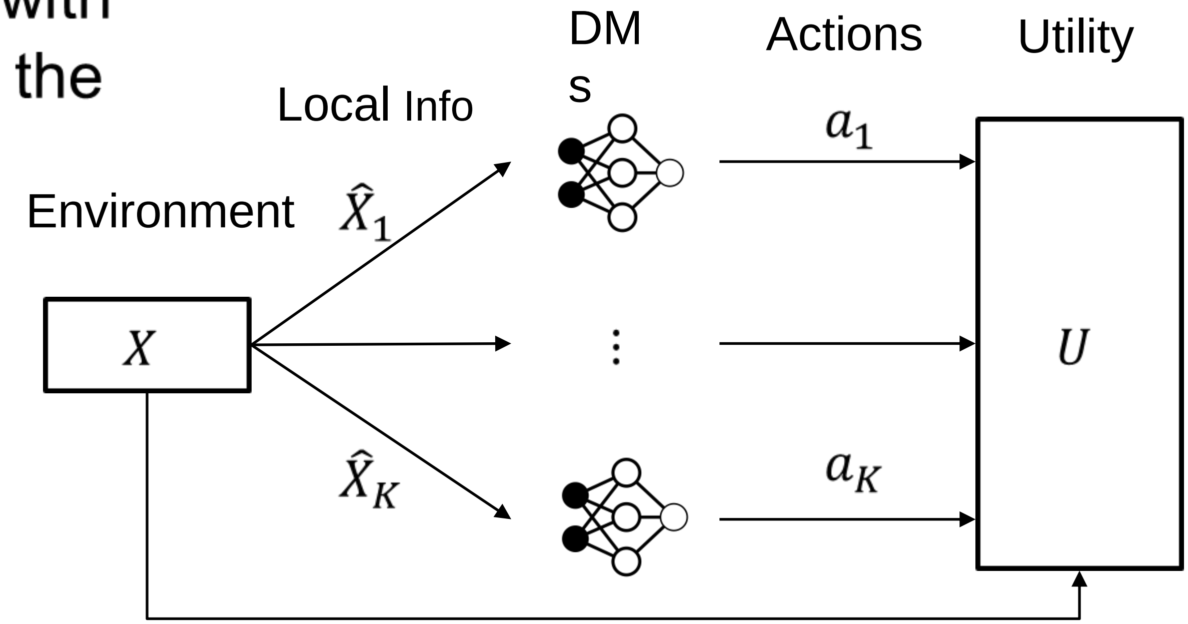
Team-Deep Neural Network:

- **Deep Neural Networks (DNNs) to represent policies.**
- **Jointly train DNNs to employ back-propagation.**

Team Deep Neural Networks

Use Deep Neural Networks (DNNs) to recast the TD problem into a parametric optimization problem, where parameters are DNNs weights.

Denote the policies with $f_{\theta_i}(\widehat{X}_i)$ where θ_i are the DNN parameters



Team DNN

Note that gradient ascent requires

$$\theta_i^{(t+1)} \leftarrow \theta_i^{(t)} + \eta_t \frac{\partial U(X, a_1, \dots, a_k)}{\partial \theta_i}$$

- **where**

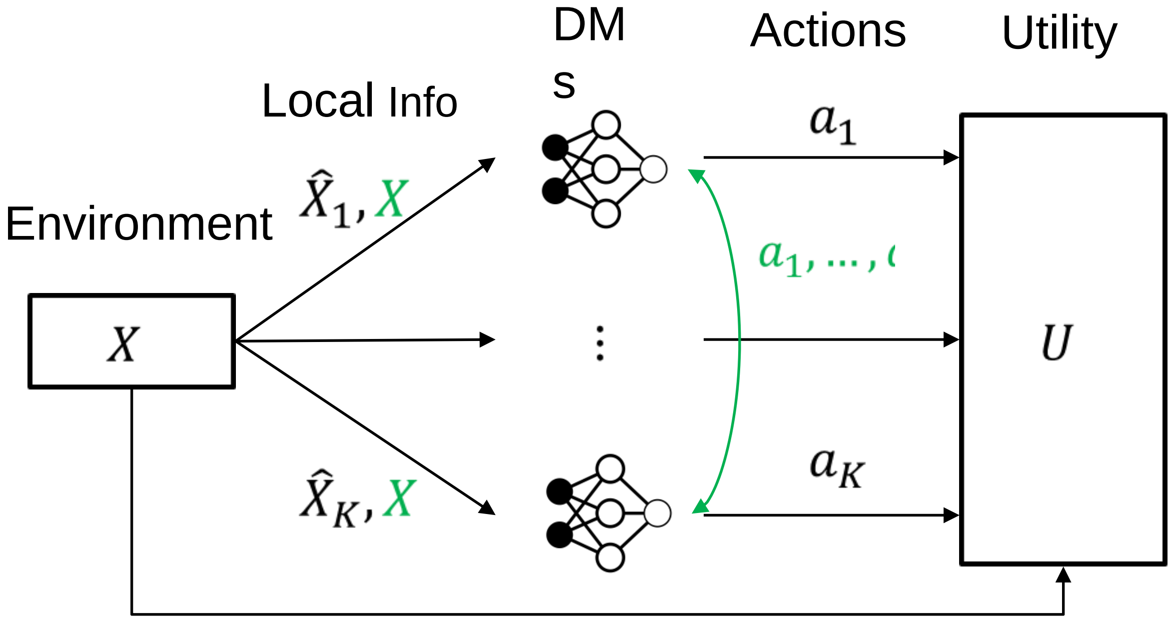
$$\frac{\partial U(X, a_1, \dots, a_k)}{\partial \theta_i} = \frac{\partial U(X, a_1, \dots, a_k)}{\partial a_i} \frac{\partial a_i}{\partial \theta_i}$$

back-propagation demands additional info (X, a_{-i}) at **user** i.

Team DNN

CENTRALIZED TRAINING:

Links and variables in green are available only during the training process



DNNs can then be jointly trained using back-propagation

Centralized Training\Decentralized Testing

Team-DNN policy design:

- **Centralized Training: given a training set** D sampled from $P_{X,\hat{X}_1,\ldots,\hat{X}_K}$ use gradient ascent to find a local maximum **of the empirical utility**

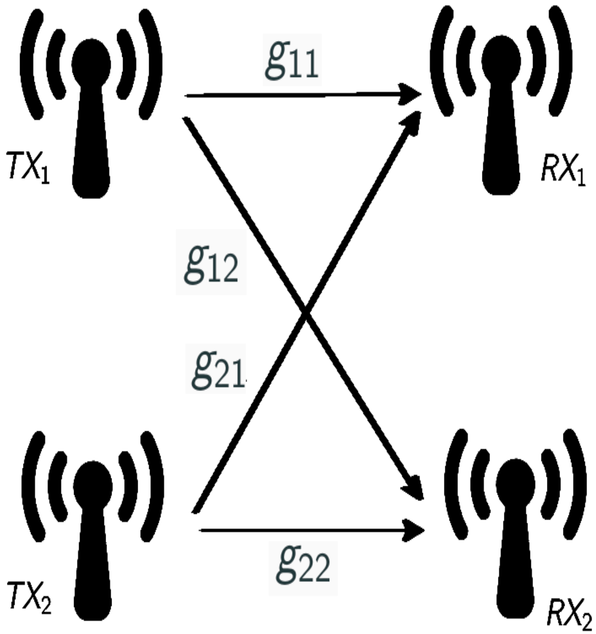
- $$\hat{U}(\theta_1, \ldots, \theta_K) = \sum_{(x,\hat{x}_1,\ldots,\hat{x}_K)} U(x, f_{\theta_1}(\hat{x}_1), \ldots, f_{\theta_K}(\hat{x}_K))$$

- **Decentralized Testing: Each DM** i **uses the DNN** $f_{\theta_i}(\hat{X}_i)$ **to map local observations** $\hat{X}_i$ **into action** a_i

Intuitive Example: Distributed Power Control

Two user SISO interference channel with fixed CSI quality

- **DMs: transmitters**
- **Environment: channel gain matrix**



$$G = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}$$

- **Local info: CSI estimates**

$$\widehat{G}_i = \sqrt{1-\sigma_i^2}G + \sigma_i \Delta_i, \qquad \sigma_i \in [0,1]$$

Δ_i is the noise (uncertainty) component in local estimates

An Example: Distributed Power Control

- **Action policies: power control algorithm**

$$\pi_i: \widehat{G}_i \to P_i \in [0, P_{max}]$$

- **Utility Function: Sum-rate**

$$U(G, P_1, P_2) = \log_2\left(1 + \frac{g_{1,1}P_1}{1 + g_{2,1}P_2}\right) + \log_2\left(1 + \frac{g_{2,2}P_1}{1 + g_{1,2}P_2}\right)$$

- **Goal:**

$$(\pi_1^*, \pi_2^*) = argmax_{\pi_1,\pi_2} E_{P_{G,\hat{G}_1,\hat{G}_2}}\left[U\left(G, \pi_1(\hat{G}_1), \pi_2(\hat{G}_2)\right)\right]$$

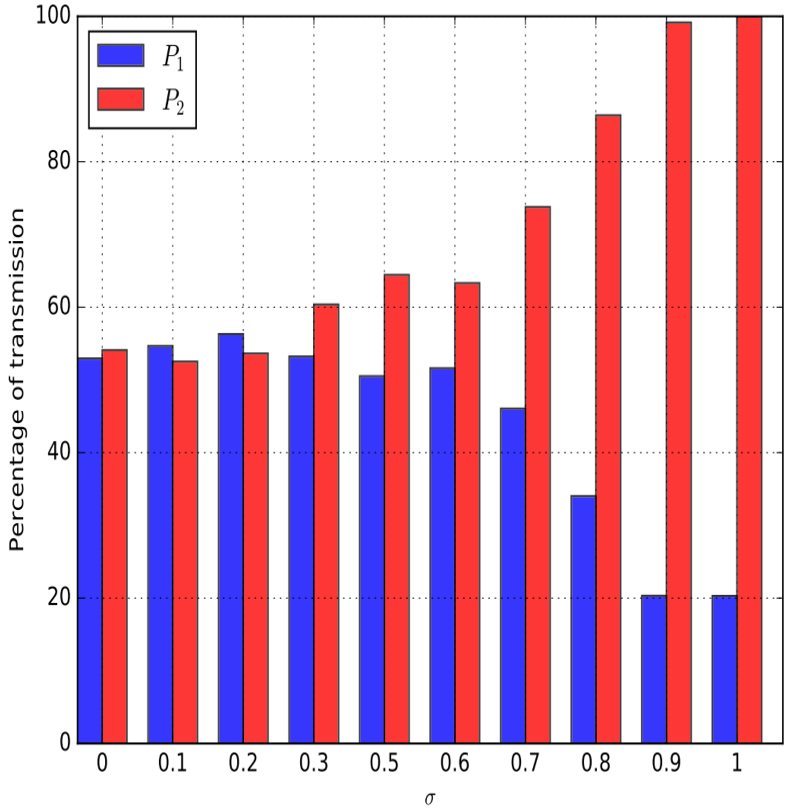
An Example: Distributed Power Control

- **For different CSI quality level (σ_1, σ_2), the Team-DNNs converge to different power control algorithms**

"Master-Slave" example with $\sigma \in [0,1]$

$$\widehat{G_1} = \sqrt{1-\sigma^2}G + \sigma\Delta$$

$$\widehat{G_2} = G$$



T-DNNs drawback

- The T-DNNs solution assumes $P_{X,\hat{X}_1,\ldots,\hat{X}_K}$ is fixed in time.

- In wireless environments, the noise affecting local observation is linked to time-varying processes (speed,positioning,…)

PROBLEM: DNNs have to be frequently retrained in order to match the current testing distribution.

Proposed solution

- Local observations are usually noisy version of the real environment state

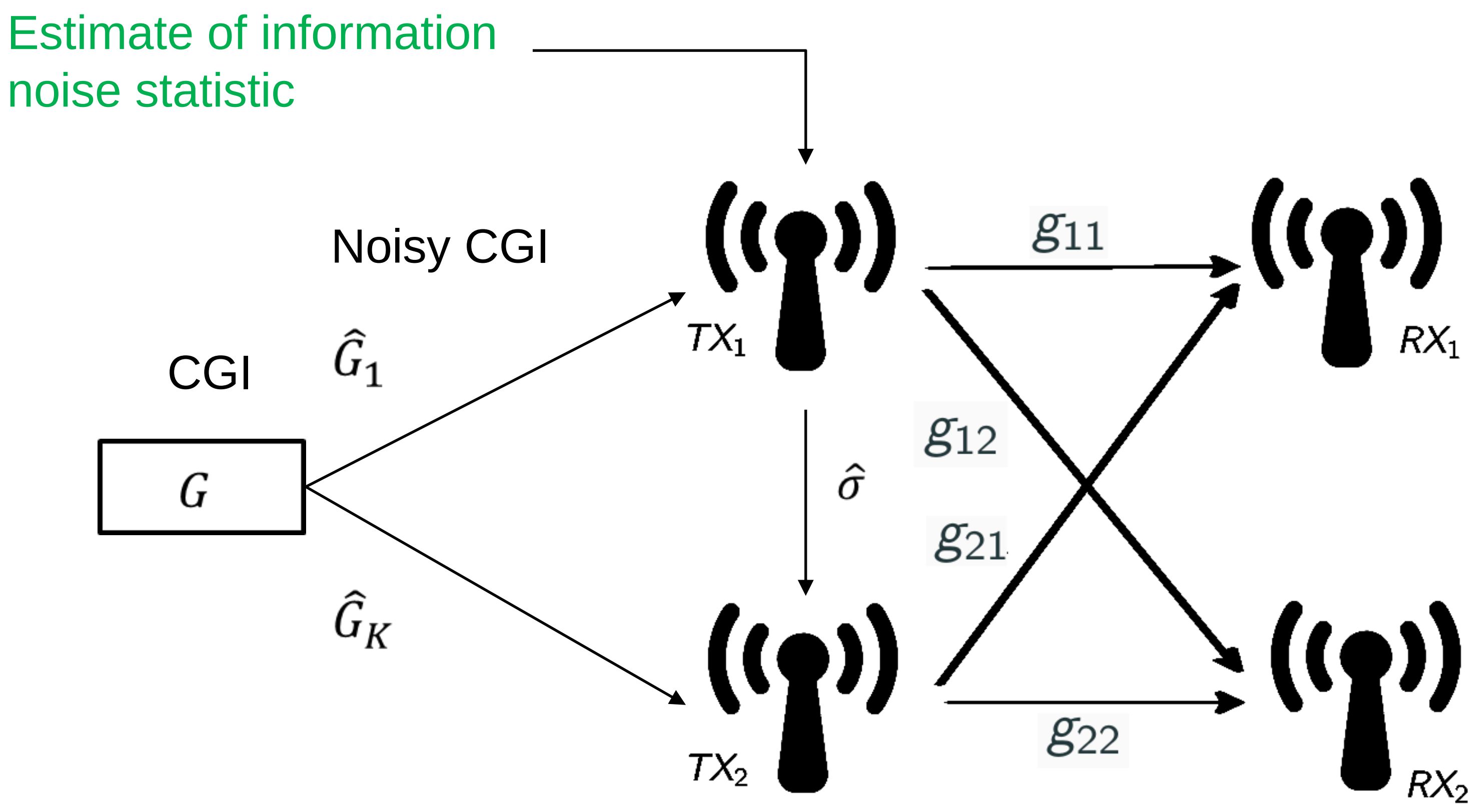
$$\hat{X}_i = f_{\sigma_i}(X)$$

where σ_i is a statistic of the noise that we assume can be estimated.

- $\vec{\sigma} = (\sigma_1, \dots, \sigma_K)$ defines the current noise scenario.

IDEA: Train a model on a multitude of noise scenarios that uses the current noise estimate to adapt its behavior,

Interference channel with noise statistics

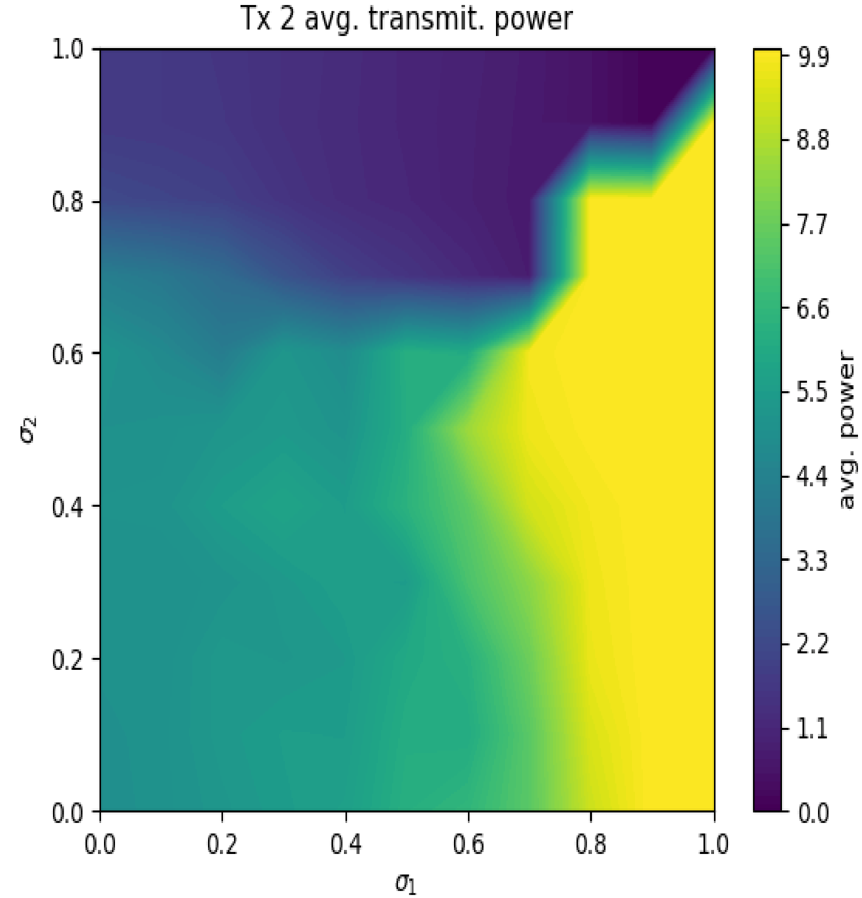


The noise statistics are assumed to be estimated separately

Interference channel with noise statistics

For any noise scenario $\vec{\sigma}$, the set of DNNs should approximate the optimal distributed power control algorithm for the specific joint distribution induced by $\vec{\sigma}$.

- Power control policies are heterogeneous over the uncertainty space (σ_1, σ_2)
- Becomes desirable having different local models specialized in different noise regimes



Mixture of Experts model

Mixture of experts (MoE):

- Ensemble learning model based on the “dividi et impera” principle
- Combines different experts (simple learning models) specialized in different parts of the input space.
- A gating network is used to properly assign experts to different input space regions.

Benefits:

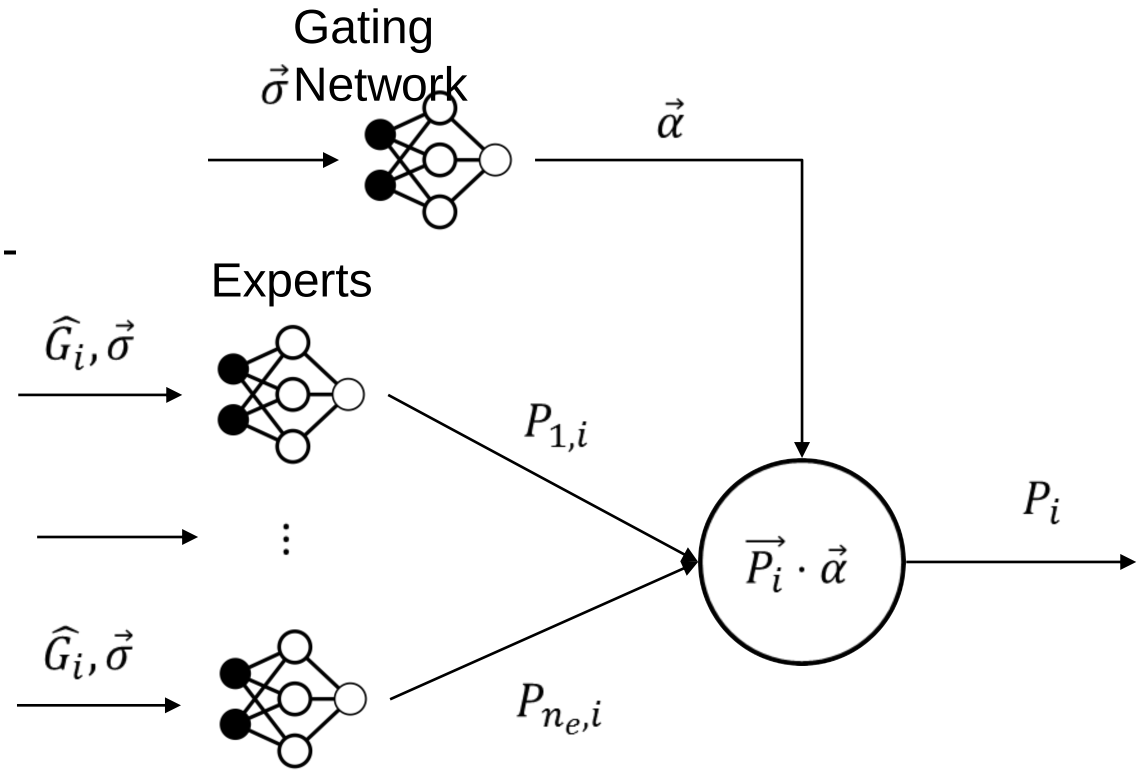
- Simpler models converge faster and are less prone to over-fitting
- Local experts can approximate different power control policies

Team Mixture of Experts

Use a Mixture of Experts (MoE) to realize the power control algorithm at DMs in order to capture the heterogeneity of the optimal power control policies

Concurrent training:

- Each expert maximizes the sum-rate optimizing its power policy on a specific region of the input space
- The gating network assigns the best expert to each noise configuration



Experiments Setup

- Two user SISO interference Rayleigh fading channel with varying CSI quality.

- Local information model for user i

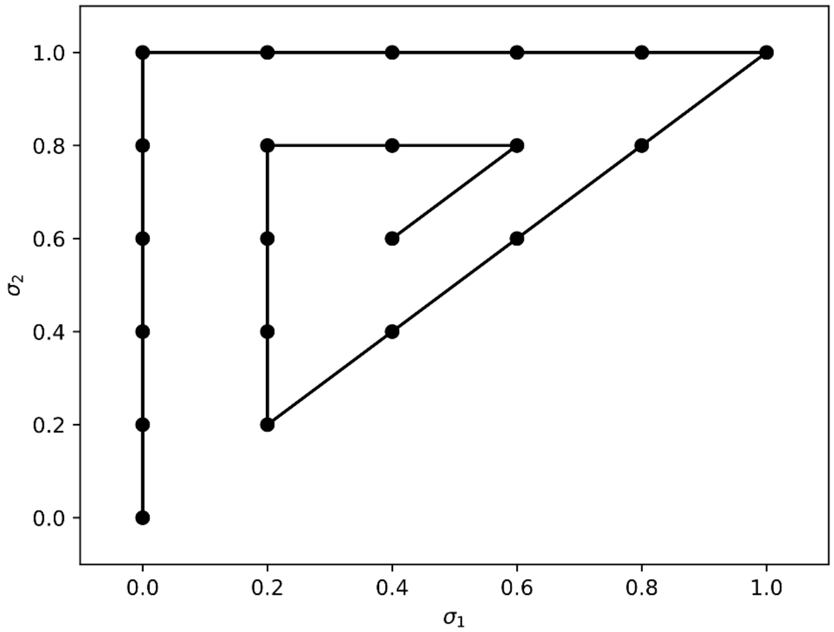
$$\hat{G}_i = \sqrt{1-\sigma_i^2}G + \sigma_i \Delta_i, \qquad \sigma_i \in [0,1]$$

- Parameters σ_i are linked to the uncertainty in the local information and we assume that can be estimated for both TXs.

Experiments Setup

- For every time-slot we compute the average sum-rate of different power control algorithms.
- Evaluate the performance of the schemes in various noise regimes and quantify the impairment due to retraining.

Information Quality Trajectory



Terms of comparison

Classical power control:

- **Perfect CGI:** optimal control scheme with perfect CSI.
- **Naïve WMSEE:** WMSEE algorithm ran with local noisy info.
- **TDMA:** One TX active.

Data-driven power control:

- **Team-MoE:** policies at DMs are realized with MoEs and are jointly trained during a single centralized phase over a multitude of noise scenarios.
- **Team-DNN**: Multi-layer perceptrons are used to represent policies at DMs and are re-trained when the noise scenario changes.

Training Phase

Team-MoE

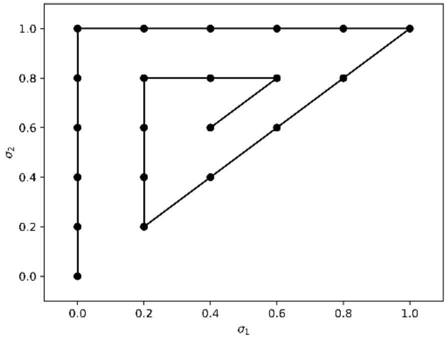
- Single training
- Data-set size: 100k data samples for various noise setting
- Batch size: 1k
- 8k gradient updates

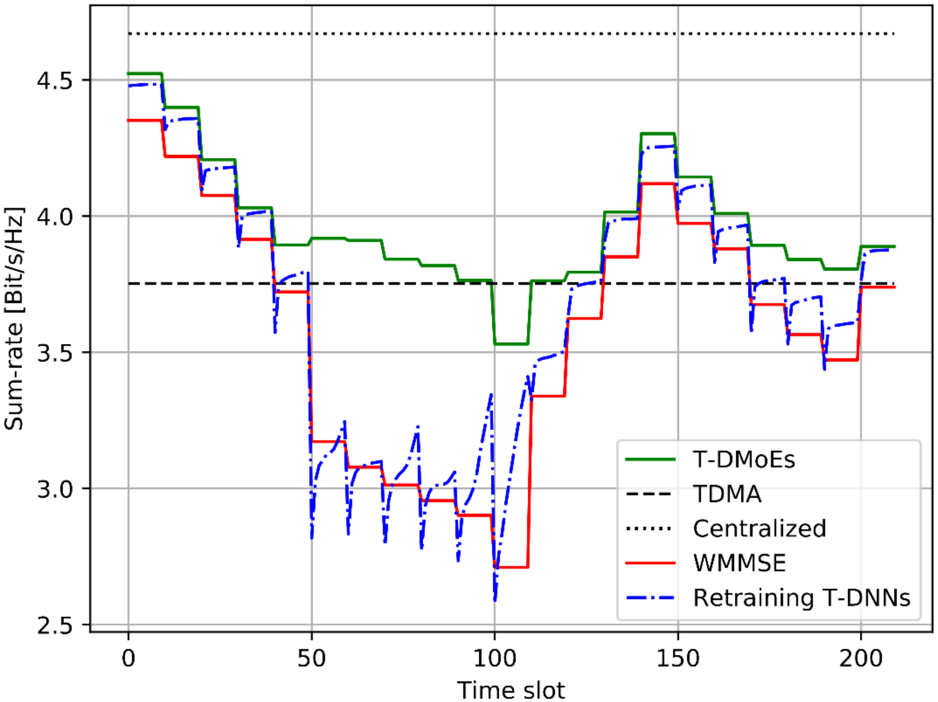
Team-DNN

- Multiple re-trainings
- Data-set size: 30k data samples from the current noise scenario (enough to have convergence)
- Batch size: 1k
- R_{up} gradient updates during each time-slot

$R_{up} \propto$ computational power available during the re-training phase.

Results $R_{up} = 10$

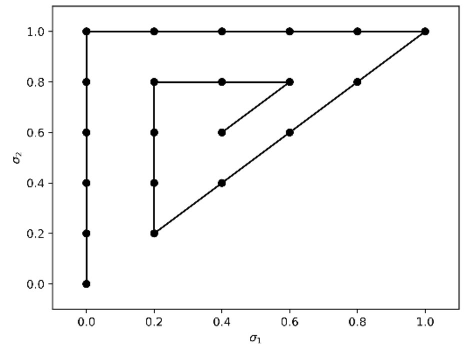


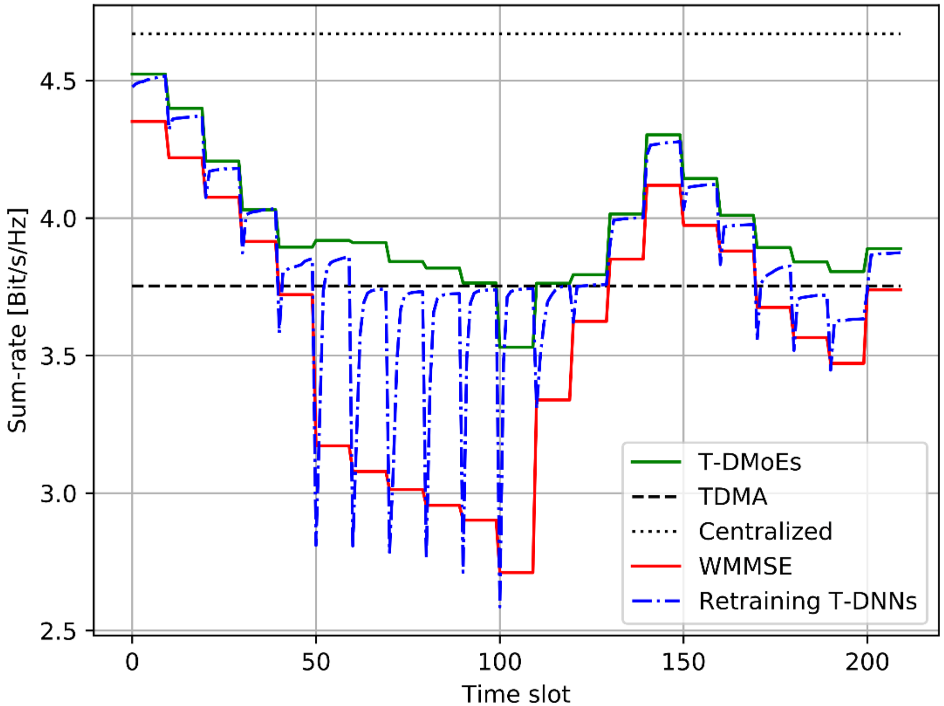


- Team-DMoE delivers highest sum-rate for almost every CGI noise configuration
- Retraining T-DNN performance are impaired by the learning process
- $R_{up} = 10$ is not enough to have convergence in a useful time

Results $R_{up} = 100$

What if increase the computational power to $R_{up} = 100$?
100 batch iteration every time-slot





The retrained T-DNN converges to the T-DMoE performance in most of the cases.

T-DMoE is learning the optimal power control policy

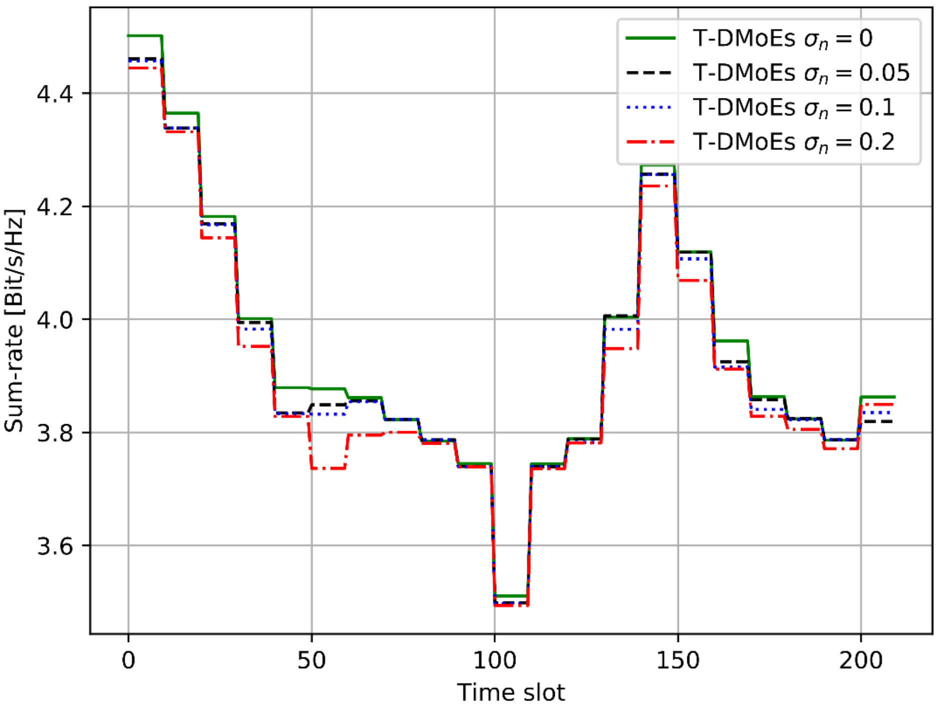
Noisy estimates

Imperfect estimates

$$\hat{\sigma} = \sigma + Z$$

Where Z is a Gaussian r.v. with zero mean and variance σ_n

Graceful degradation of sum-rate as estimates get worse



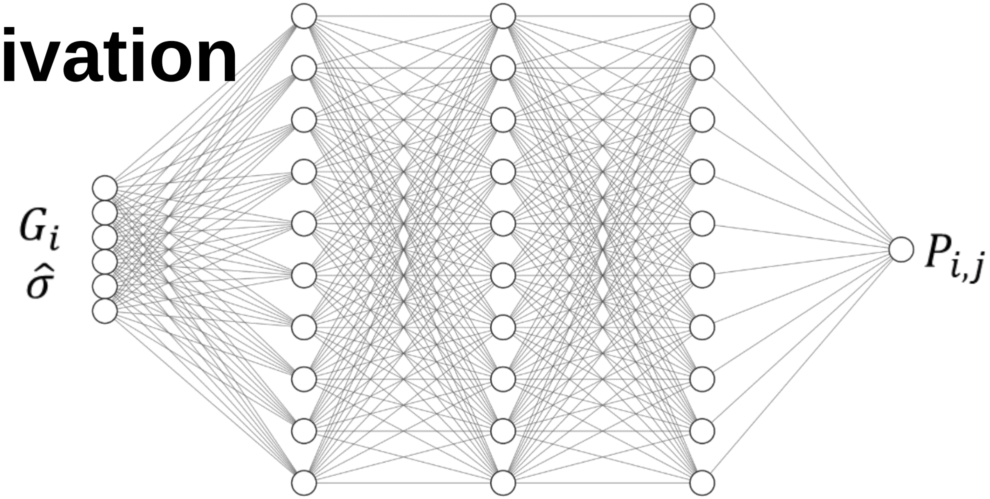
Conclusions

- Team-DNN can learn optimally robust decentralized policies under arbitrary uncertainties^[1,2]
- Team-DNN need to estimate the amount and structure of uncertainty
- Centralized Training/Decentralized Retraining requires burdensome and frequent retraining if noise statistics info are not employed
- By exploiting noise statistics estimates, an “universal” model can be trained using Mixture of Experts^[3]
- Extension: finite-rate message making DNNs to exchange relevant info among agents before decision^[4]

- [1] P. de Kerret, D. Gesbert, M. Filippone, "Decentralized Deep Scheduling for Interference Channels", in Proceedings of the IEEE Workshop on Machine Learning in Communications Systems, workshop of the International Conference on Communications (ICC), 2018, Kansas City, Mo. USA.
- [2] D. Gunduz, P. de Kerret, C. Murthy, D. Gesbert, M. van der Schaar, N.D. Sidiropoulos, "Machine Learning in the Air", in IEEE Journal Selected Areas in Communications, September 2019 (also on arxiv https://arxiv.org/abs/1904.12385).
- *[3] M. Zecchin, D. Gesbert, M. Kountouris, "Team Deep Mixture of Experts for Distributed Power Control", in proc. IEEE Signal Processing Advances for Wireless Communications Workshop, Atlanta, 2020*
- [4] M. Kim, P. de Kerret, D. Gesbert, "Team Deep Learning for Decentralized Optimization in Wireless Communications" in Proc. Of IEEE BalkanCom 2019.

Experts

- **3 hidden layers**
- **10 neurons/layer**
- **ReLu activation**



Gating Network

- Input: uncertainty estimates
- Structure: Fully connected with 2 hidden layers, 10 neurons and ReLu activations
- Output: Softmax activation to obtain a weighting vector for experts selection