# Deep Learning Applications in Astronomy

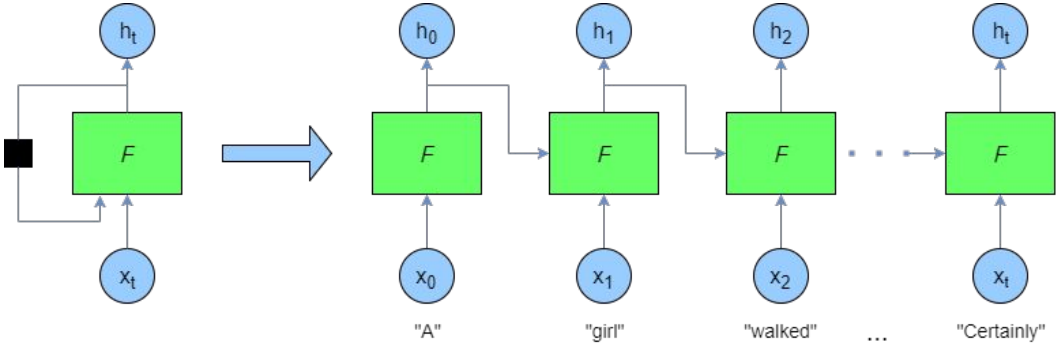
**Clécio R. Bom**

**clearnightsrthebest.com**

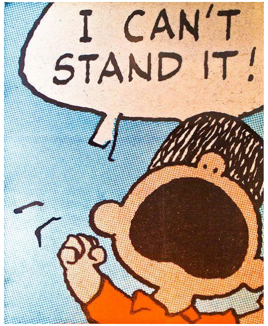
debom@cbpf.br

**February 14, 2019**

# Recurrent Neural Nets




But .. there is the gradient vanishing problem ….

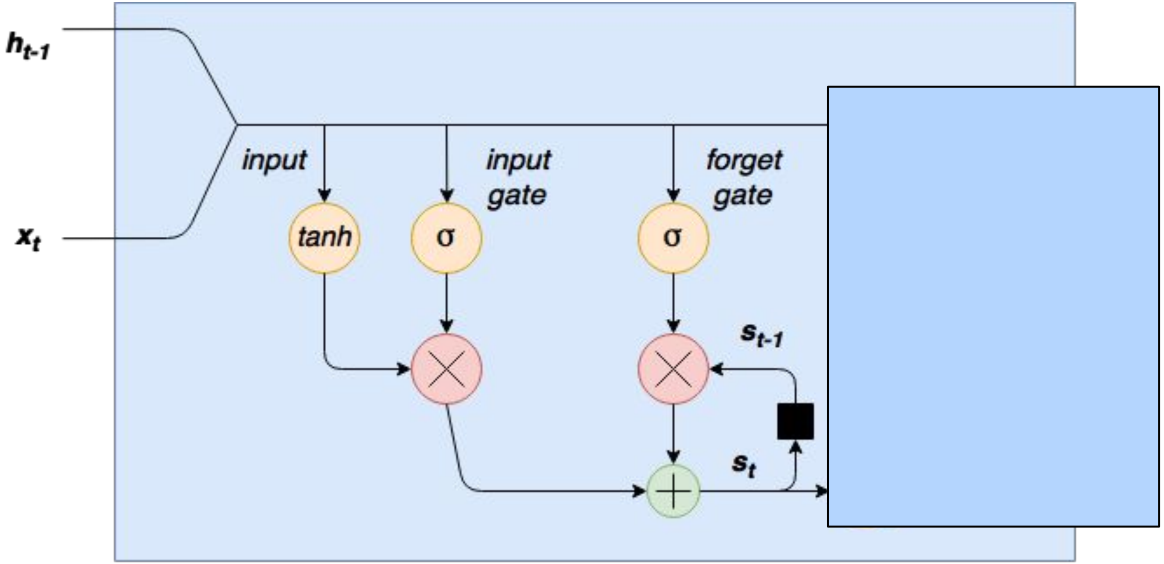



Long term correlations ….

Brazil is a great ________

"I have been staying in Brazil for the last 20 years. I can speak fluent ________."

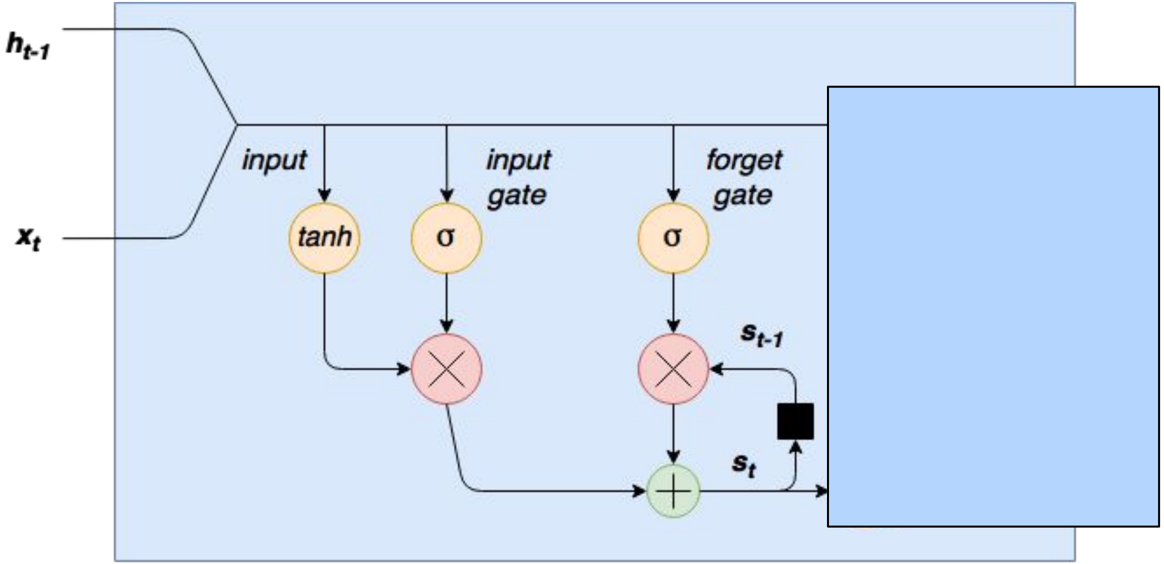
# LSTM Cell




$$f = \sigma(w_f x_t + U_k h_{t-1}) \quad k = \sigma(w_k x_t + U_k h_{t-1})$$

$$i = \Phi(w_i x_t + U_i h_{t-1})$$

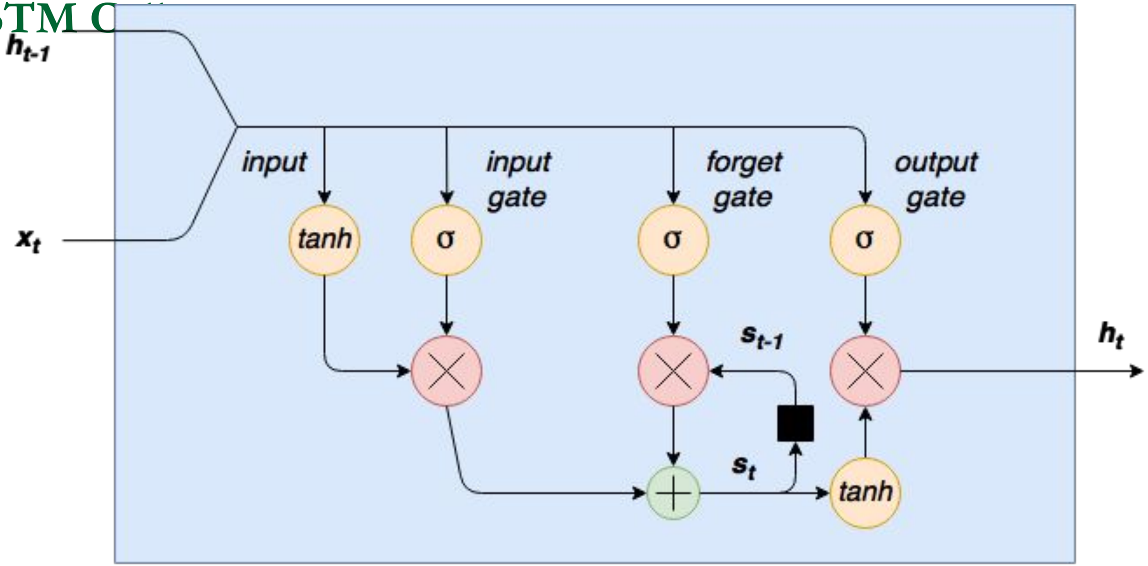
# LSTM Cell




$$f = \sigma(w_f x_t + U_k h_{t-1}) \quad k = \sigma(w_k x_t + U_k h_{t-1})$$

$$s_t = s_{t-1} \circ f + k \circ i$$

LSTM C




$$f = \sigma(w_f x_t + U_k h_{t-1}) \quad k = \sigma(w_k x_t + U_k h_{t-1})$$

$$s_t = s_{t-1} \circ f + k \circ i$$

# LSTM – A couple of considerations

- It helps to prevent vanishing gradient, however it did not solve it completely….
- Long sequences problems, Try hundreds not thousands
- More effective in forecasting than classification
- It made **history** with chatbots, translators, speech-to-text, etc…
- It might need lot of embedding ….
- Resource Expensive if compared to Resnets…

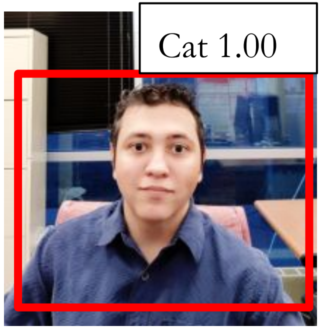
# LSTM – Long sequences Strategy

- Truncate
- Embed
- Subsample
- Auto Encoders

# How sure is my NN ?

In a Deep Learning classification problem that classifies dogs and cats would classify a human as a dog or a cat anyway. It would not be possible to know that the image is not a dog, neither  a cat.

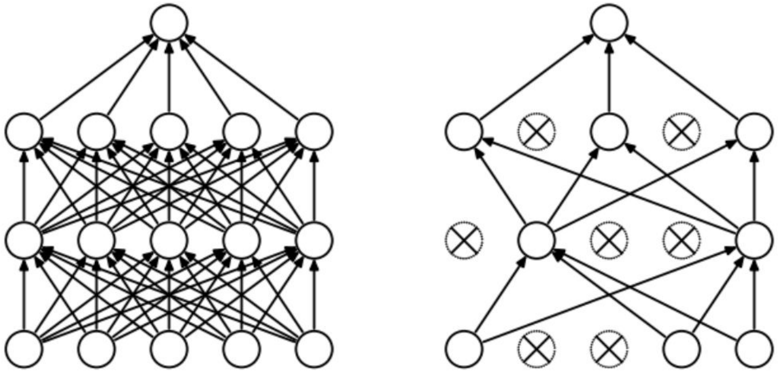
It would be interesting to have a framework in which one could derive a PDF on the predictions ….



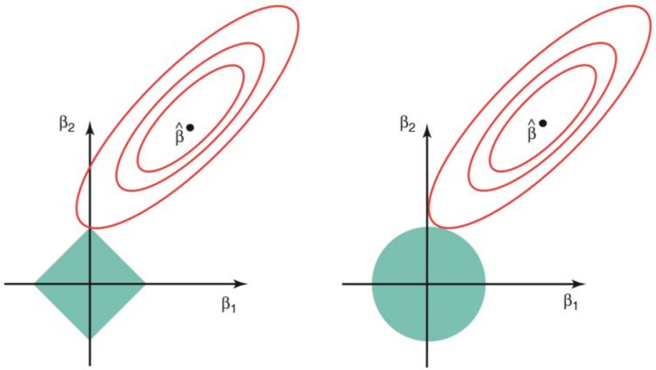
# Bayesian Nets

One may think in the weights initialization as a definition of a prior p(w). That is if we randomly initializes it would be like a flat prior. If we add a regularization technique, that would define a different prior.

When using dropout in the forward-pass (or any other stochastic regularisation technique), a randomly drawn masked weight matrix corresponds to a function draw.



(a) Standard Neural Net (b) After applying dropout.



# How sure is my NN ?

Three perspectives for uncertainty.
Epistemic uncertainty  quantifies our ignorance about the models most suitable to explain our data. Aleatoric uncertainty captures noise inherent in the environment. The Predictive uncertainty conveys the model's uncertainty in its output.

The dropout probability, together with the weight configuration of the network, determine the magnitude of the epistemic uncertainty.

For a fixed dropout probability p, high magnitude weights will result in higher output variance, i.e. higher epistemic uncertainty. With a fixed p, a model wanting to decrease its epistemic uncertainty will have to reduce its weight magnitude (and set the weights to be exactly zero to have zero epistemic uncertainty).

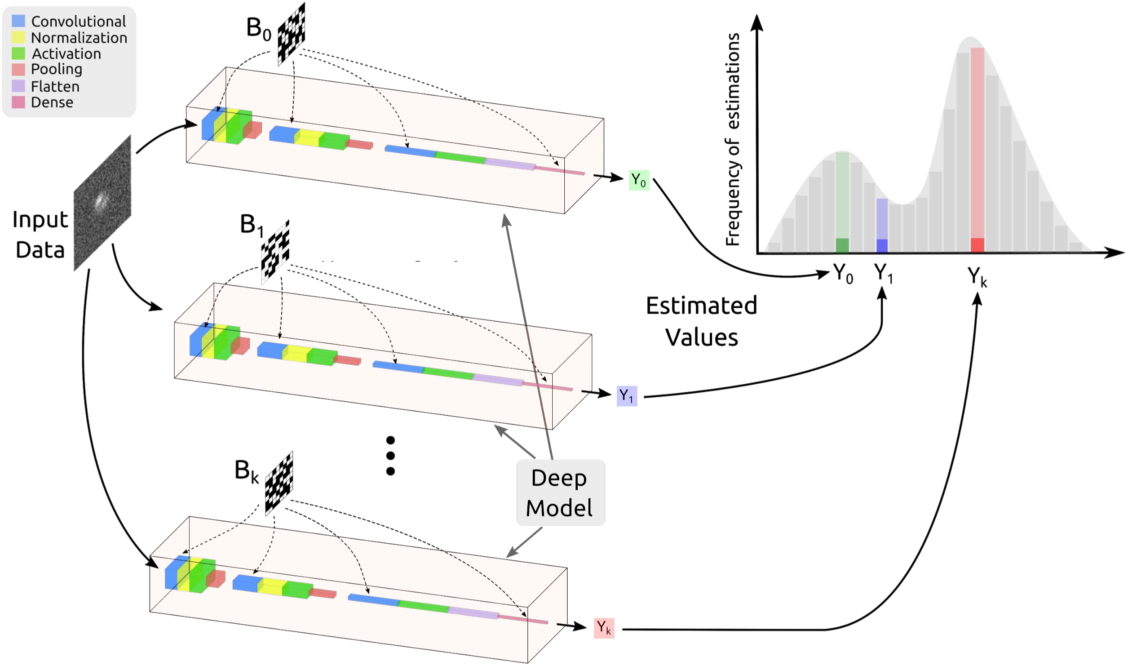
arXiv:1705.07832

arXiv:1711.00165

Lee, Jaehoon, et al. "Deep neural networks as gaussian processes." *arXiv preprint arXiv:1711.00165* (2017)

# Error estimative

One way out is to perform a grid-search in dropout probabilities to minimize the epistemic uncertainty. However this is computationally very expensive.




Or the dropout probability can be optimised using a gradient method ...

# Bayesian Nets - Variational

According to the variational interpretation, dropout is seen as an approximating distribution p(ω|X,Y) to the posterior in a Bayesian neural network with a set of random weight matrices ω = {Wl} L l=1 with L layers and θ the set of variational parameters.

Consider the weights ω and a training dataset X = {x1,...,xN} and the corresponding outputs Y = {y1,...,yN}, the posterior of the network weights, p(ω|X,Y), captures the plausible network parameters. With this posterior, we can calculate the probability distribution of the values of an output y for a new test input point x by marginalizing over all possible weights ω

$$p(\mathbf{y}|\mathbf{x},\mathbf{X},\mathbf{Y}) = \int p(\mathbf{y}|\mathbf{x},\omega)\, p(\omega|\mathbf{X},\mathbf{Y})\, d\omega\,.$$

With p(ω|X,Y), we can calculate the probability distribution of the values of an output y for a new test input point x by marginalizing over all possible weights ω.

# Bayesian Nets - Variational

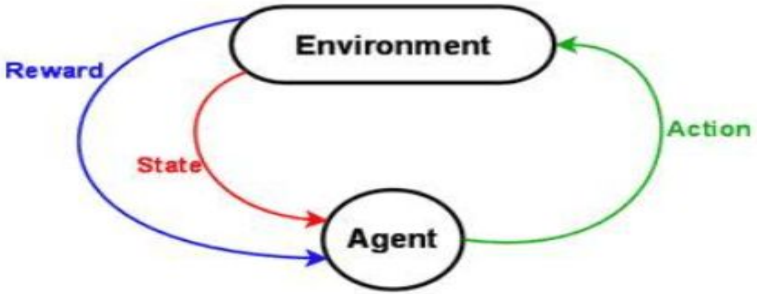
Consider the weights ω and a training dataset X = {x1,...,xN} and the corresponding outputs Y = {y1,...,yN}, the posterior of the network weights, p(ω|X,Y), captures the plausible network parameters. With this posterior, we can calculate the probability distribution of the values of an output y for a new test input point x by marginalizing over all possible weights ω

$$p(\mathbf{y}|\mathbf{x}) \approx \int p(\mathbf{y}|\mathbf{x}, \omega)\, q(\omega)\, d\omega$$

We consider an approximating variational distribution, q(ω), with an analytic form. The parameters of q(ω) are then optimized to transform q(ω) to be as close as possible to the true posterior.
This is performed by adding minimizing their Kullback-Leibler (KL) divergence between.

# Reinforcement Learning

Nor supervised, not unsupervised.. learning by experience.




$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$$

A Markov decision process relies on the Markov assumption, that the probability of the next state si+1 depends only on current state si and action ai, but not on preceding states or actions.

# Reinforcement Learning

## Playing Atari with Deep Reinforcement Learning

**Volodymyr Mnih** **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

**Daan Wierstra** **Martin Riedmiller**

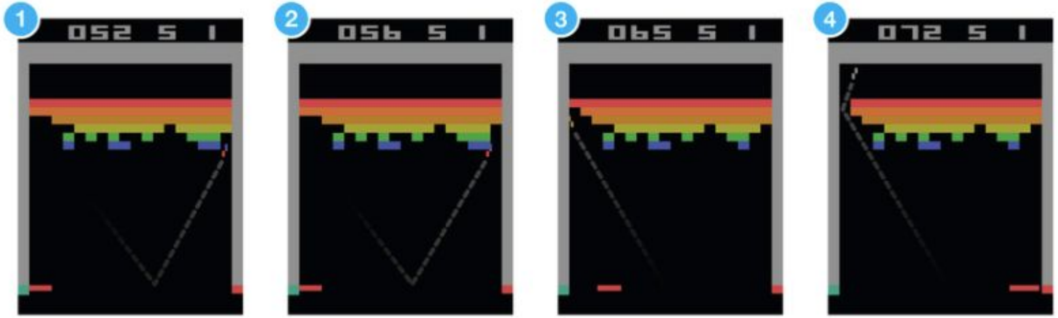
DeepMind Technologies

arXiv:1312.5602

Figure 1: Screen shots from five Atari 2600 Games: (*Left-to-right*) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

# Reinforcement Learning

Play video Games!




Example adapted from: https://ai.intel.com/demystifying-deep-reinforcement-learning/

# Reinforcement Learning

Total Reward

$$R = r_1 + r_2 + r_3 + \ldots + r_n$$

Future Reward

$$R_t = r_t + r_{t+1} + r_{t+2} + \ldots + r_n$$

Discounted Future Reward

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \ldots + \gamma^{n-t} r_n$$

$$R_t = r_t + \gamma(r_{t+1} + \gamma(r_{t+2} + \ldots)) = r_t + \gamma R_{t+1}$$

Example adapted from: https://ai.intel.com/demystifying-deep-reinforcement-learning/

# Reinforcement Learning

If we set the discount factor γ=0, then our strategy will be short-sighted and we rely only on the immediate rewards. If we want to balance between immediate and future rewards, we should set discount factor to something like γ=0.9. If our environment is deterministic and the same actions always result in same rewards, then we can set discount factor γ=1.

A good strategy for an agent would be to always choose an action that maximizes the (discounted) future reward.

Discounted Future Reward

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \ldots + \gamma^{n-t} r_n$$

$$R_t = r_t + \gamma(r_{t+1} + \gamma(r_{t+2} + \ldots)) = r_t + \gamma R_{t+1}$$

Example adapted from: https://ai.intel.com/demystifying-deep-reinforcement-learning/

# Q Learning

The maximum discounted future reward when we perform action a in state s, and continue optimally from that point on

The best possible score at the end of the game after performing action a in state s

$$Q(s_t, a_t) = max\ R_{t+1}$$

Bellman equation

$$Q(s,a) = r + \gamma max_{a'} Q(s', a')$$

maximum future reward for this state and action is the immediate reward plus maximum future reward for the next state

Example adapted from: https://ai.intel.com/demystifying-deep-reinforcement-learning/

# Q Learning

$$L = \frac{1}{2}[\underbrace{r + max_{a'}Q(s', a')}_{\text{target}} - \underbrace{Q(s, a)}_{\text{prediction}}]^2$$

```
initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
     select and carry out an action a
     observe reward r and new state s'
     Q[s,a] = Q[s,a] + α(r + γ max_a' Q[s',a'] - Q[s,a])
     s = s'
until terminated
```

Example adapted from: https://ai.intel.com/demystifying-deep-reinforcement-learning/

# Q Learning

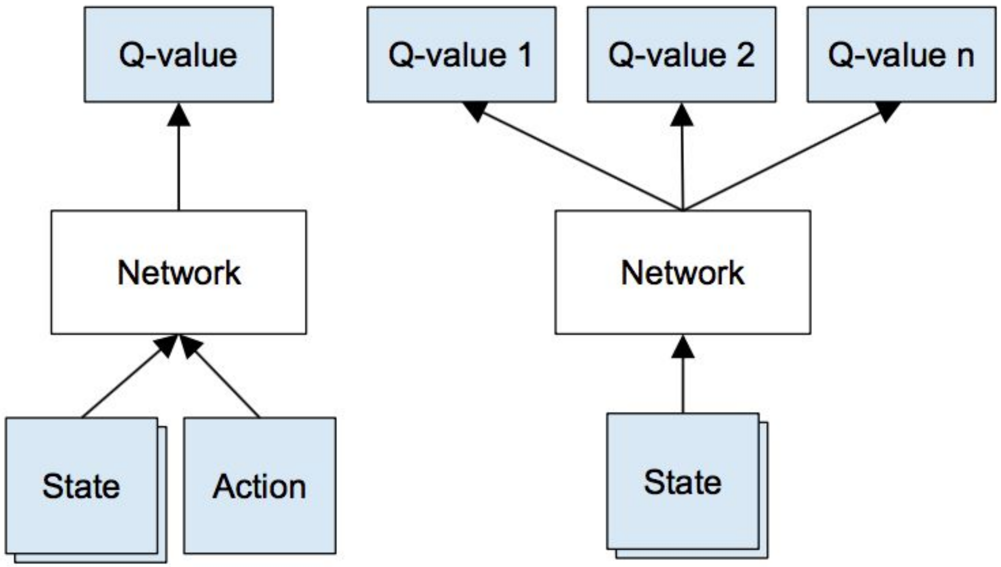
$$L = \frac{1}{2}[\underbrace{r + max_{a'}Q(s', a')}_{\text{target}} - \underbrace{Q(s, a)}_{\text{prediction}}]^2$$

```
def q_learning_with_table(env, num_episodes=500):
    q_table = np.zeros((5, 2))
    y = 0.95
    lr = 0.8
    for i in range(num_episodes):
        s = env.reset()
        done = False
        while not done:
            if np.sum(q_table[s,:]) == 0:
                # make a random selection of actions
                a = np.random.randint(0, 2)
            else:
                # select the action with largest q value in state s
                a = np.argmax(q_table[s, :])
            new_s, r, done, _ = env.step(a)
            q_table[s, a] += r + lr*(y*np.max(q_table[new_s, :]) - q_table[s, a])
            s = new_s
    return q_table
```

Example adapted from: https://ai.intel.com/demystifying-deep-reinforcement-learning/

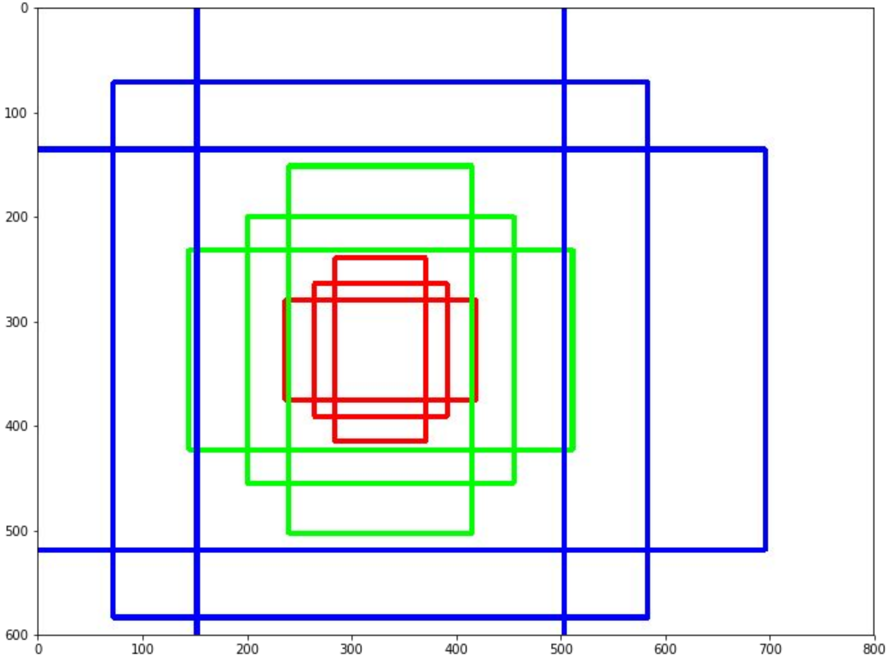
# Reinforcement Learning

Play video Games!

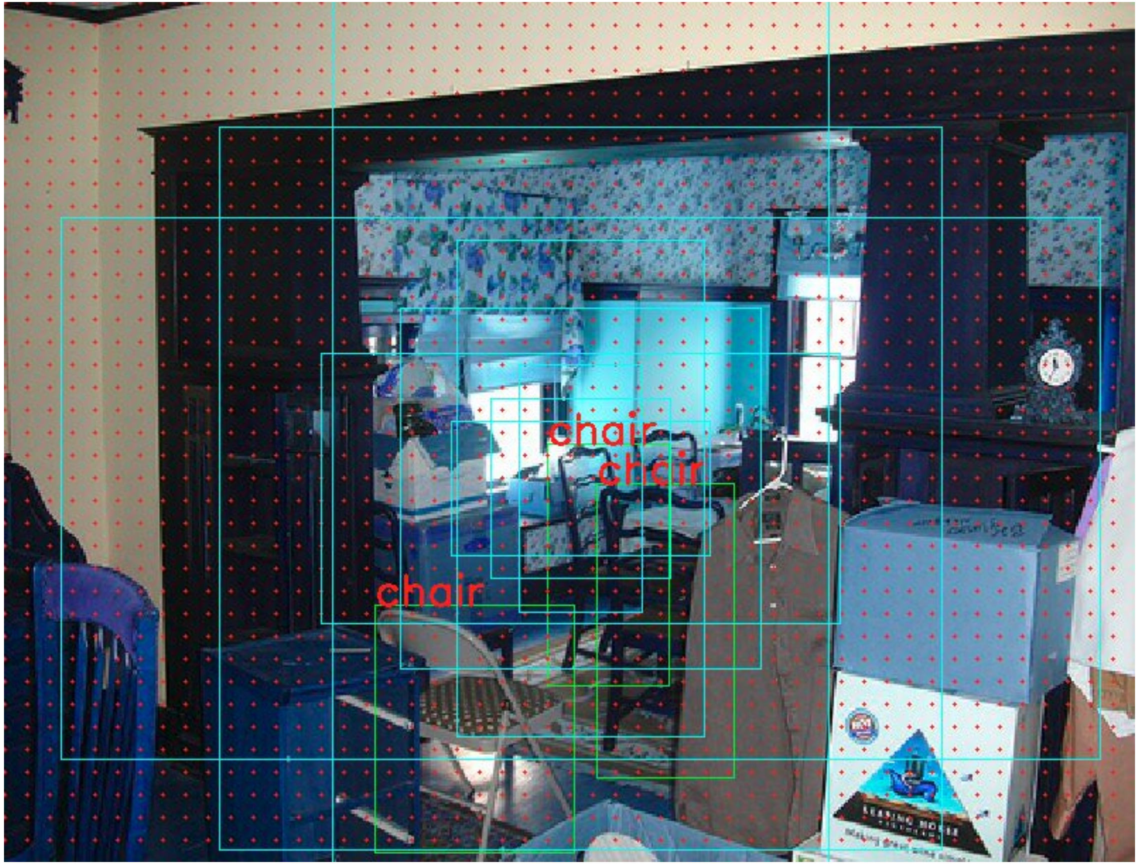



We could also take only game states input and output the Q-value for each possible action.

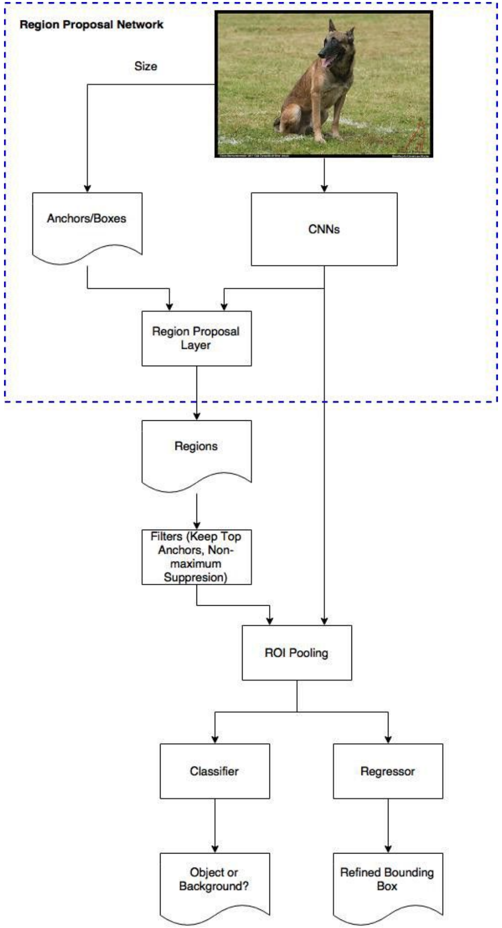
# Region Proposal Neural Networks



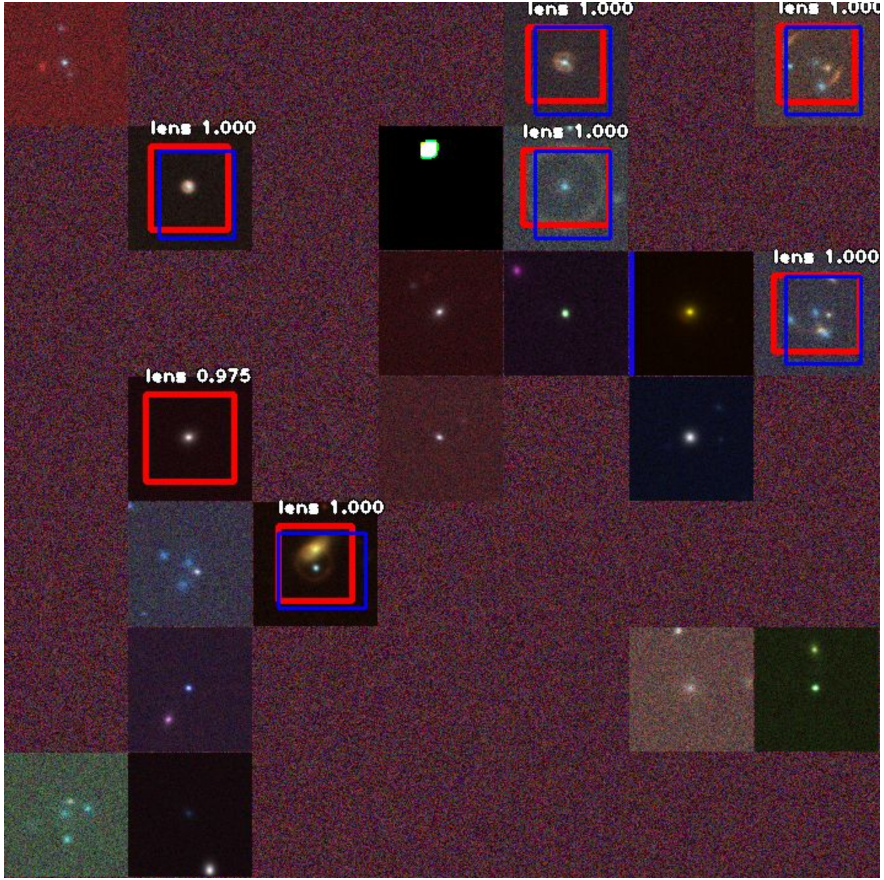
# Region Proposal Neural Networks



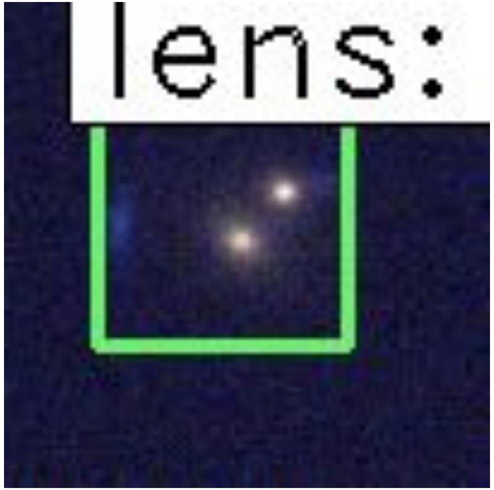
# Region Proposal Neural Networks



# Region Proposal Neural Networks




RetinaNet Architecture (2017) adapted.



# What people are doing with this?

UNCERTAINTIES IN PARAMETERS ESTIMATED WITH NEURAL NETWORKS: APPLICATION TO STRONG GRAVITATIONAL LENSING

LAURENCE PERREAULT LEVASSEUR, YASHAR D. HEZAVEH*, AND RISA H. WECHSLER
Kavli Institute for Particle Astrophysics and Cosmology, Stanford University, Stanford, CA, USA
SLAC National Accelerator Laboratory, Menlo Park, CA, 94305, USA

*Draft version August 30, 2017*

arXiv:1708.08843

DEEP RECURRENT NEURAL NETWORKS FOR SUPERNOVAE CLASSIFICATION

TOM CHARNOCK[1] AND ADAM MOSS[1]

[1] *School of Physics & Astronomy*
*University of Nottingham, Nottingham, NG7 2RD, England*

arXiv:1606.07442

**Radio Galaxy Zoo: ClaRAN — a deep learning classifier for radio morphologies**

Chen Wu[1★], O. Ivy Wong[1†], Lawrence Rudnick[2], Stanislav S. Shabala[3]
Matthew J. Alger[4,5], Julie K. Banfield[4,6], Cheng Soon Ong[5], Sarah V. White[7],
Avery F. Garon[2], Ray P. Norris[8,9], Heinz Andernach[10], Jean Tate[11]
Vesna Lukic[12], Hongming Tang[13], Kevin Schawinski[14], Foivos I. Diakogiannis[15,1]

# Radio Galaxy Zoo: ClaRAN — a deep learning classifier for radio morphologies

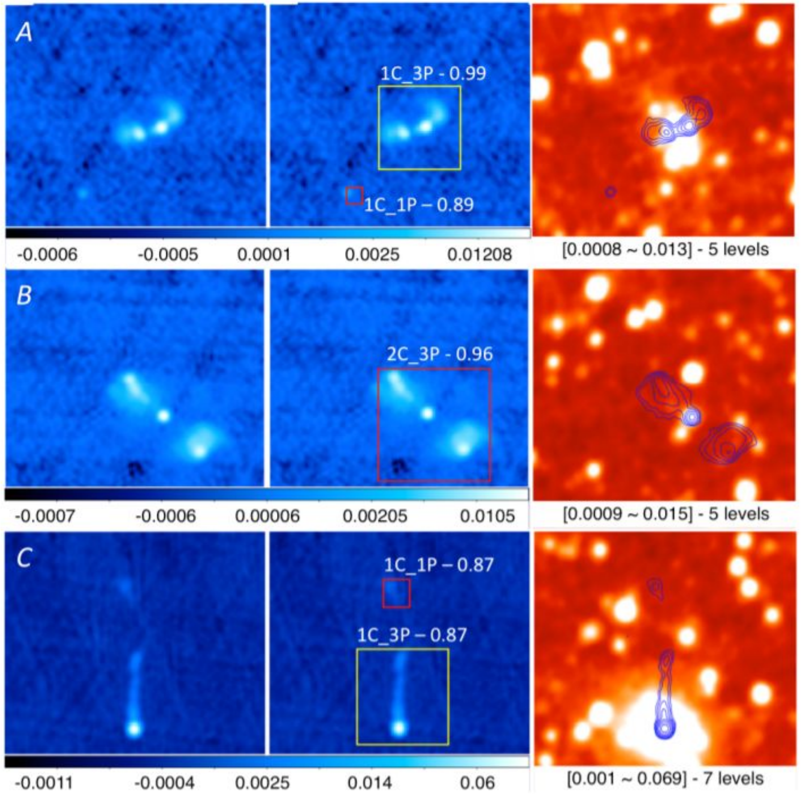



**Figure 1.** Three classification examples (*A*, *B*, and *C*) on RGZ subjects — each of them 3' × 3' in size — `FIRST J081700.6+571626`, `FIRST J070822.2+414905`, and `FIRST J083915.7+285125`. The first column shows the FIRST radio emission. The second column shows the ClaRAN output — a box encompassing each identified source, and its morphology is labelled as $i$**C**_$j$**P**, where $i$ and $j$ denotes the number of radio components and the number of radio peaks respectively. Each morphology label is associated with a score between 0 and 1, indicating the probability of the quoted morphology class. The first two columns share the same color bar at the bottom, denoting flux density values in Jy/beam. The last column shows the corresponding WISE infrared image overlaid with $5\sigma$ radio contours. The contour levels (Jy/beam) are shown at the bottom of each infrared image.

# Radio Galaxy Zoo: ClaRAN — a deep learning classifier for radio morphologies

First, for each selected subject $f$, we ensure *all* radio sources within $f$ have a user-weighted Consensus Level (CL) no less than 0.6. CL measures the relative agreement levels of classification among citizen scientists and is defined in Banfield et al. (2015) as the largest fraction of the total classifications for a radio source that have been agreed upon. This is to ensure most radio sources exposed to ClaRAN are morphologically human-resolvable.

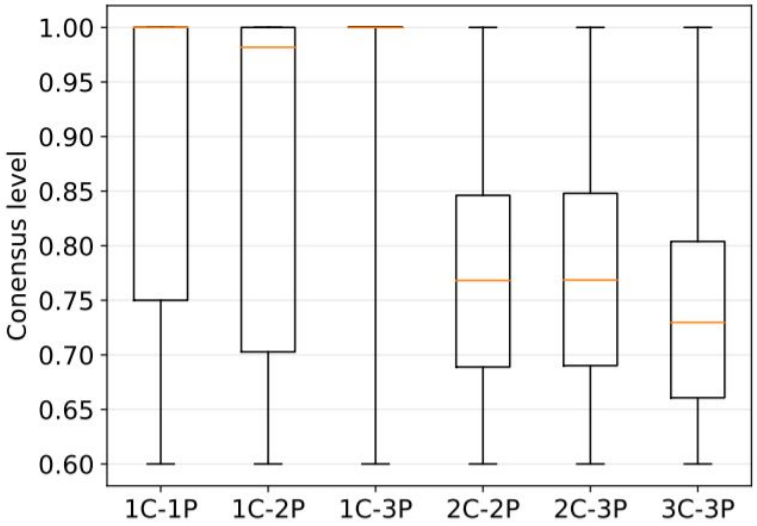
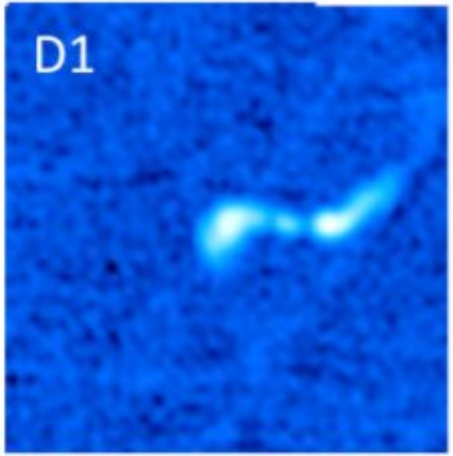
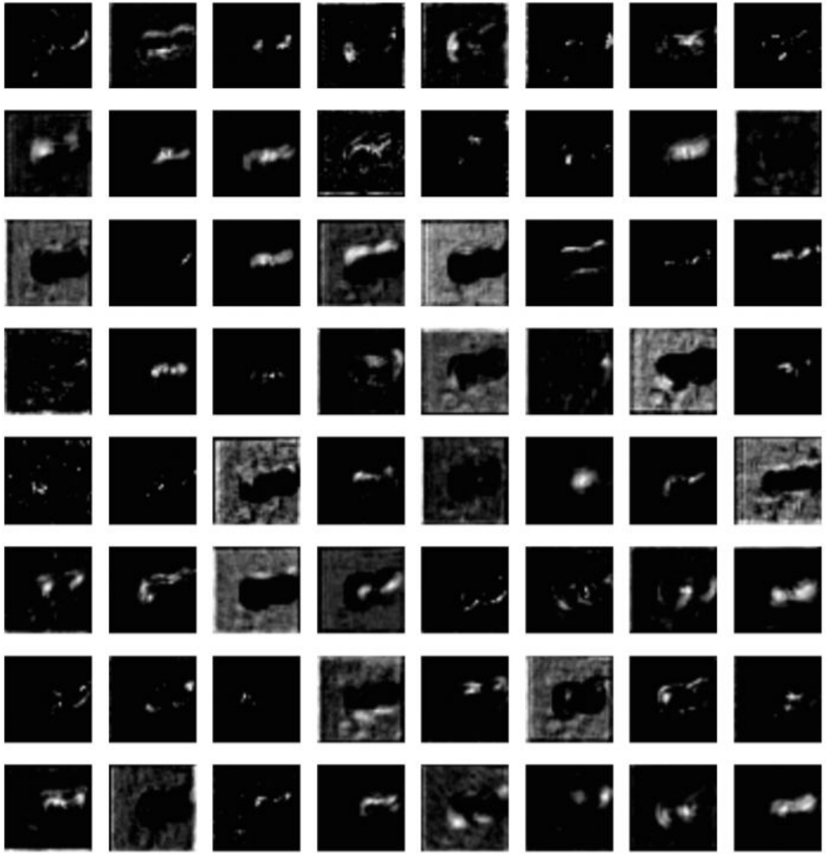


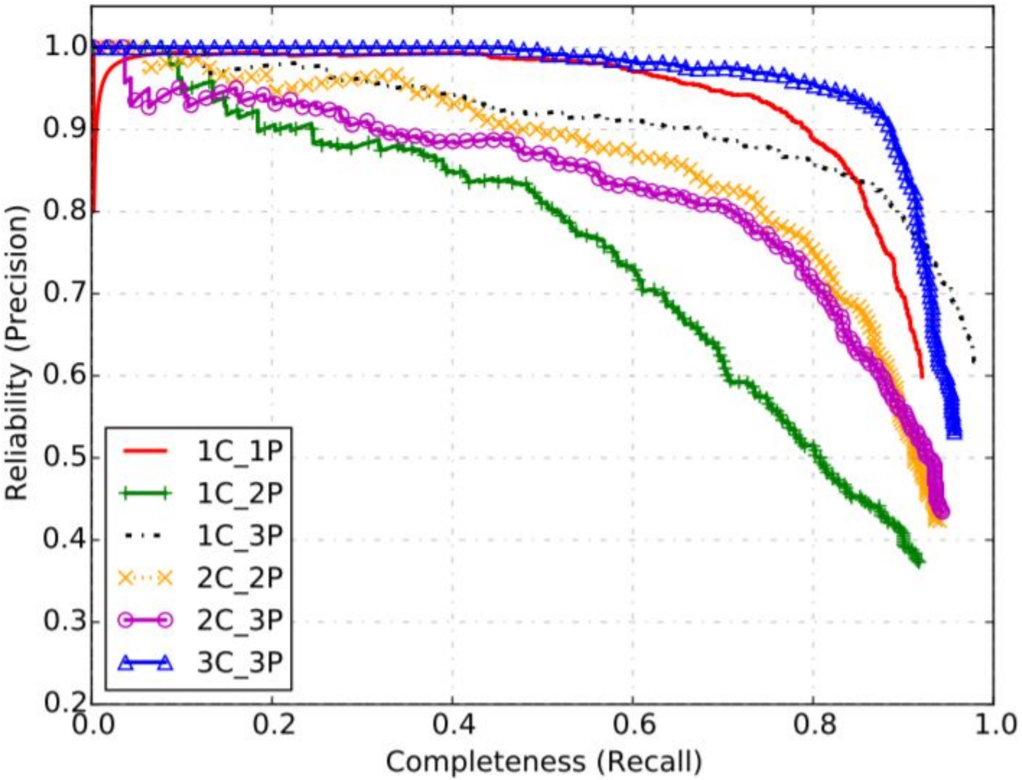

**Figure 2.** The distribution of the consensus level (CL) across six morphology classes in the data set that consists of 10,744 RGZ subjects selected from DR1. The whiskers above and below the

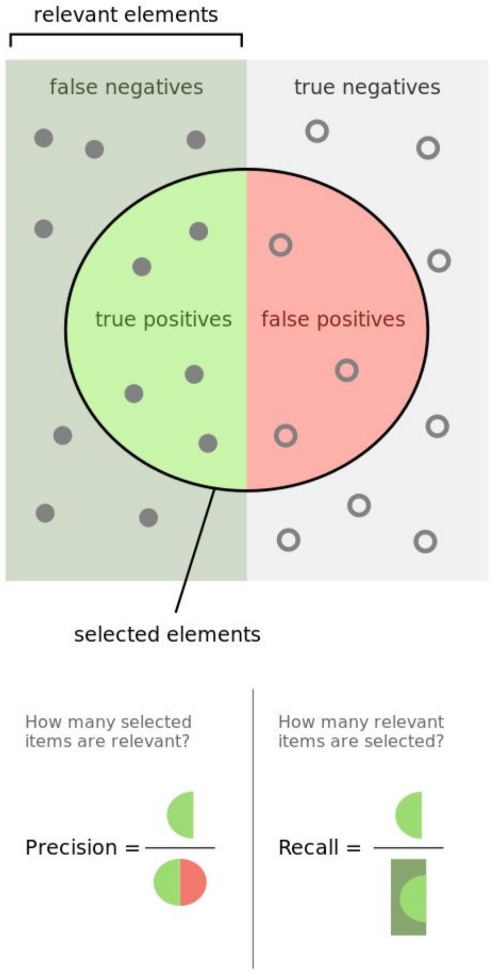
# Radio Galaxy Zoo: ClaRAN — a deep learning classifier for radio morphologies



## Radio Galaxy Zoo: ClaRAN — a deep learning classifier for radio morphologies



# DEEP RECURRENT NEURAL NETWORKS FOR SUPERNOVAE CLASSIFICATION

Tom Charnock[1] and Adam Moss[1]

| Time | g | r | i | z |
|---|---|---|---|---|
| $t_1$ | $g_1$ | $r_1$ | $i_1$ | $z_1$ |
| $t_2$ | $g_2$ | $r_2$ | — | $z_2$ |
| $t_3$ | $g_3$ | $r_3$ | $i_3$ | $z_3$ |

**Table 1.** Data augmentation of missing observations. The missing data is replaced randomly by a value between $i_1$ and $i_3$.

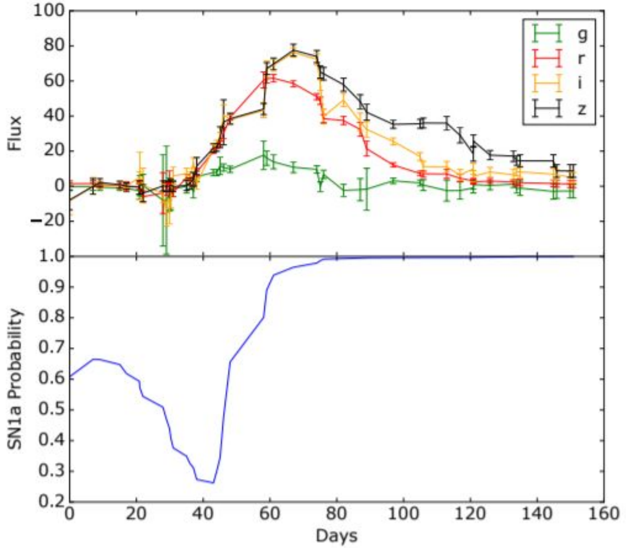



**Figure 3.** (Top) Example light curve in the 4 $g, r, i, z$ bands for SN ID 551675 (a type-Ia) in the Supernovae Photometric Classification Challenge data Kessler et al. (2010a).

# UNCERTAINTIES IN PARAMETERS ESTIMATED WITH NEURAL NETWORKS: APPLICATION TO STRONG GRAVITATIONAL LENSING

LAURENCE PERREAULT LEVASSEUR, YASHAR D. HEZAVEH*, AND RISA H. WECHSLER
Kavli Institute for Particle Astrophysics and Cosmology, Stanford University, Stanford, CA, USA
SLAC National Accelerator Laboratory, Menlo Park, CA, 94305, USA

*Draft version August 30, 2017*

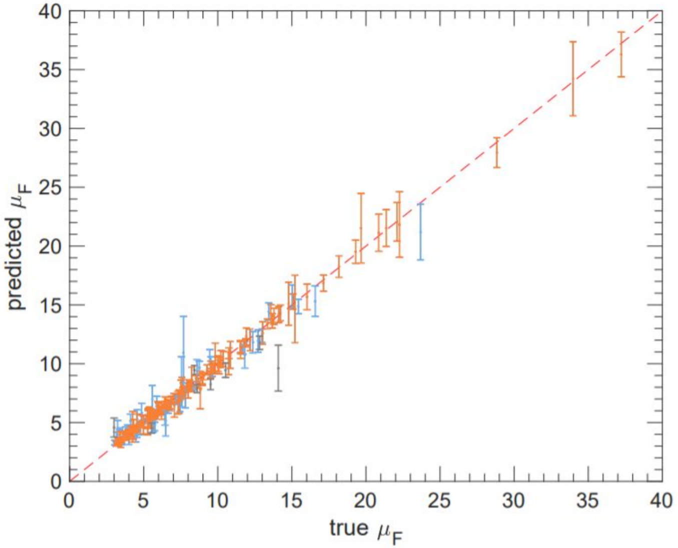
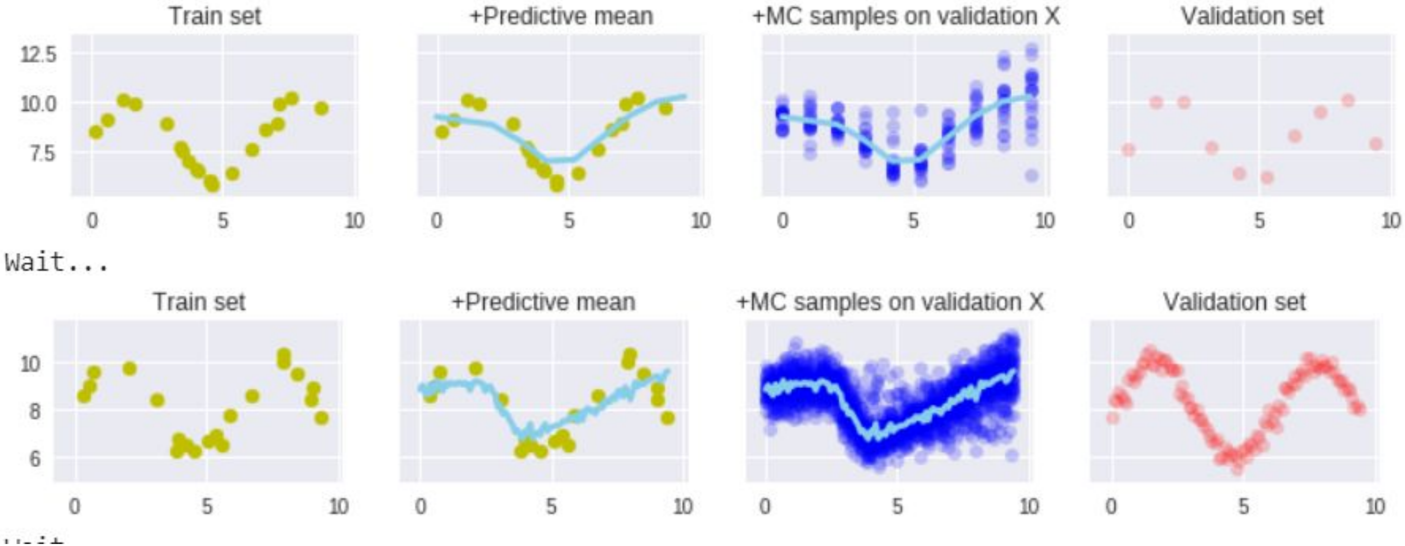



**Figure 1.** Predicted 68.3% uncertainties for lensing flux magnification, $\mu_F$, as a function of the true value of this parameter. The orange, blue, and black errorbars correspond to examples where the true values fall within the 68.3, 95.5, and 99.7% confidence intervals respectively.

# Example 04 - Concrete Dropouts

This is a regression problem in which we try to predict the sin(x) function. However we use the concrete dropout method to derive a sample of predicts measurements.



CEFET/RJ

CBPF

# Deep Learning
# Applications in Astronomy

**Clécio R. Bom**

## Thank you!
## Hope you have a deeper understanding than a few days ago.

**clearnightsrthebest.com**

debom@cbpf.br

**February 14, 2019**