Human-in-the-Loop Feature Selection

A paper by Alvaro Correia and Freddy Lecue

Presented by Joseph Allen

Outline

What Is It?

- **Human-in-the-Loop (HITL) – a model that requires human interaction.**
  - Allows for the identification of problems and requirements that may not be easily/automatically identified.
- **Feature Selection – the process of selecting a subset of relevant features (variables, predictors, etc) for use in model construction.**
  - Often done to simplify models (for explainability), shorten training times, avoid the curse of dimensionality, and enhance generalization by reducing overfitting.

Motivation

- **Clearly feature selection is useful**
- **Currently, feature selection is mostly a manual process done by consulting human experts about the most important features for a problem.**
  - But this process is time-consuming and expensive.
  - Also, the experts with this knowledge are rarely the ones building the actual machine learning models.
- **Some automatic feature selection can be done, but these methods only generate a single subset of features for all examples.**
- **Ideally, feature selection should be influenced and reinforced by user feedback**

Goals

- **Automate the feature selection process**
- **Involve model designers, domain experts, and users in the feature selection process**
- **Simplify the model by only considering a subset of the most important features**
- **Improve model performance/accuracy by utilizing feedback**
- **Provide per-example explanations of predictions**

This Paper's Approach

1) Have experts manually identify the most important features for a few examples

2) Use that feedback to model the probability of selecting each feature given an example

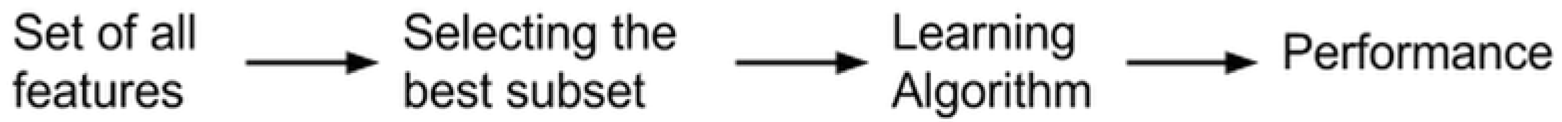
3) Derive a RL policy that produces a new feature subset for each example in the dataset.

- This policy is learned through policy gradient methods that minimize both the loss function of the learning algorithm and the dissimilarity of model-chosen and human-chosen feature subsets

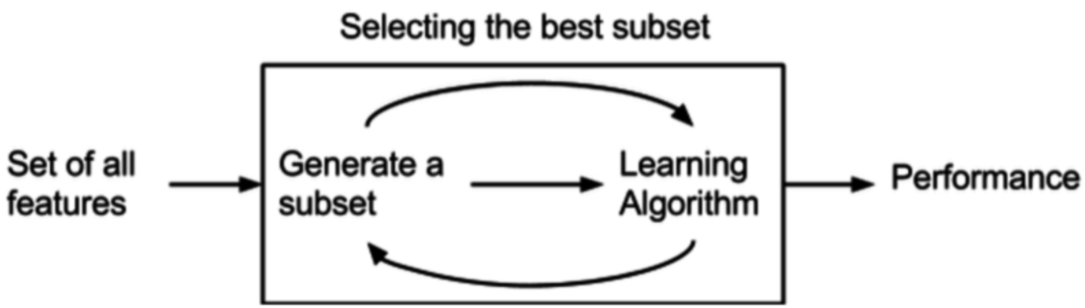
Related Work

Feature selection/elimination methods:

- <u>Filters</u> – consist of a preprocessing step where the top-k features are selected by ranking them against some score function, such as the mutual information between input and target variables (Tyagi and Mishra 2013)

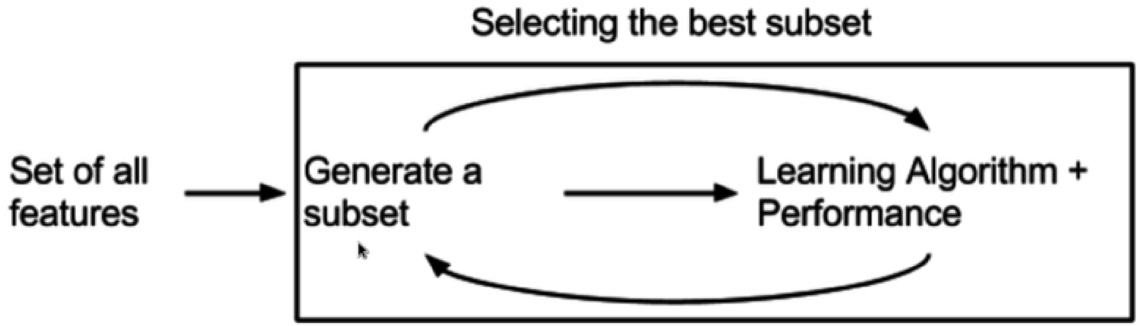


- <u>Wrappers</u> – rank feature subsets following some performance measure such as the accuracy in the training data (Kohavi and John 1997)



Related Work

- Embedded Methods – similar to wrappers, but different in that feature selection is performed while training the learning algorithm (Guyon and Elisseeff 2003)
  - The approach in this paper is an embedded method, since the feature selection and learning algorithms are jointly trained via gradient descent



  - This method reduces the significant computation time of wrapper methods, since the learning algorithm is only trained once

Related Work

Per-example Feature Selection:

- A filter-based method utilizing mutual information was proposed in (Avdiyenko, Bertschinger, and Jost 2012), however:
  - It is deterministic and completely data-driven, whereas the model in this paper includes human feedback
  - It builds feature subsets sequentially, while this paper's model does so in a single step

Attention Mechanisms:

- By attributing weights to different hidden states in a neural network, attention mechanisms are also selecting the most relevant features to minimize the model's cost function
- The work of this paper applies these ideas in HITL framework
- More info: (Xu et al. 2015; Bengio, Leonard, and Courville 2013; Bahdanau, Cho, and Bengio 2015)

Framework

- **In this framework, expert annotators provide the ground truth and select the most relevant features that influenced their decision**
  - This selection is modeled with a binary mask a that is applied element-wise to filter out the irrelevant features of an example x, where $x \in \mathbb{R}^d$ and each dimension corresponds to a feature x_j

$$\begin{cases} a_j = 1, \text{ if } x_j \text{ is used for example } x \\ a_j = 0, \text{ otherwise.} \end{cases}$$

  - The input to the learning algorithm then becomes $x' = x \odot a$
  - The cost function $C(x', y, \theta)$ must be differentiable w.r.t. the function that defines a if we hope to minimize the cost using gradient descent

Framework

- However, it is not differentiable if a is an indicator variable
- So we model the probability of a given x instead.
- We do so by associating each component $a_j \in \{0,1\}^d$ with a Bernoulli distribution parameterized by $\hat{q}_j$ and obtain:

$$\pi(a|x) = \prod_{j=1}^{d} \hat{q}_j^{a_j} (1 - \hat{q}_j)^{(1-a_j)}$$

- The goal then is to concentrate the mass of the distribution $\pi(a|x)$ on values of a that minimize the loss function $C(x', y, \theta)$
  - $\pi(a|x)$ should then tell us the best set of features given any input
  - The probability $\hat{q}_j$ of each feature being selected can be approximated by a neural network with parameters ψ :

$$\hat{q}_\psi = (\hat{q}_1, \hat{q}_2, ..., \hat{q}_d) = h(x; \psi)$$

Framework

Human-Like Feature Selection:

- We've discussed how expert feedback can be modeled and used to learn the feature selection algorithm, but the framework can also be used to mimic the feature selection of users.
  - Let's denote the user feedback by q (the learned model gives $\hat{q}$)
  - Intuitively, if either q_j or $\hat{q}_j$ are close to 1, then feature j is determinant to predict $\hat{y}$ and is negligible otherwise.
- We can define a similarity measure between q_j and $\hat{q}_j$ (which also serves as the cost function $C_f(x, q, \psi)$) to measure how well a model fits user feedback:
  - Euclidean distance: $\mathcal{C}_f(x, q, \psi) = \mathbb{E}_x \, ||q_i - \hat{q}_i||^2 = \mathbb{E}_x \, ||q_i - h(x;\psi)||^2$
  - Cosine distance: $$C_f(x, q, \psi) = 1 - \frac{q\,\hat{q}}{||q||_2\,||\hat{q}||_2} = 1 - \frac{q\,h(x;\psi)}{||q||_2\,||h(x;\psi)||_2}$$

Framework

- With the similarity measure $C_f(x, q, \psi)$ defined, we simply add it to the final cost function, which we'll denote by C:

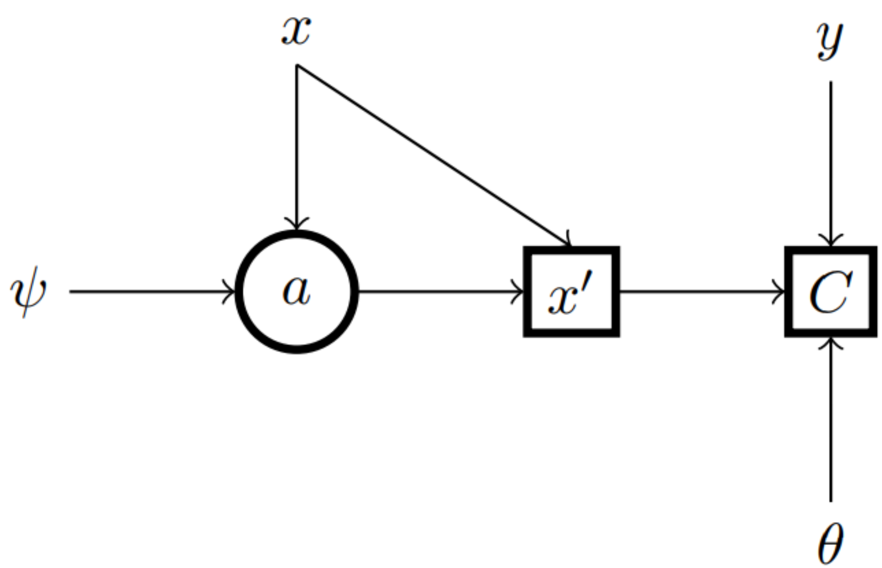
$$\mathcal{C} = \mathcal{C}(x, y, q, \theta, \psi) = \mathcal{C}(x', y, \theta) + \lambda \mathcal{C}_f(x, q, \psi)$$

  where λ is a hyperparameter that balances the tradeoff between these two signals

  - Intuitively, this cost function encourages the agent to achieve good performance at the machine learning task while mimicking human feature selection, which serves as a bias of sorts.
  - For examples with no human feedback available, $C_f(x, q, \psi) = 0$

Stochastic Computation Graphs

- a is stochastic and we need to sample it before making a prediction, so we can frame the model as a stochastic computation graph



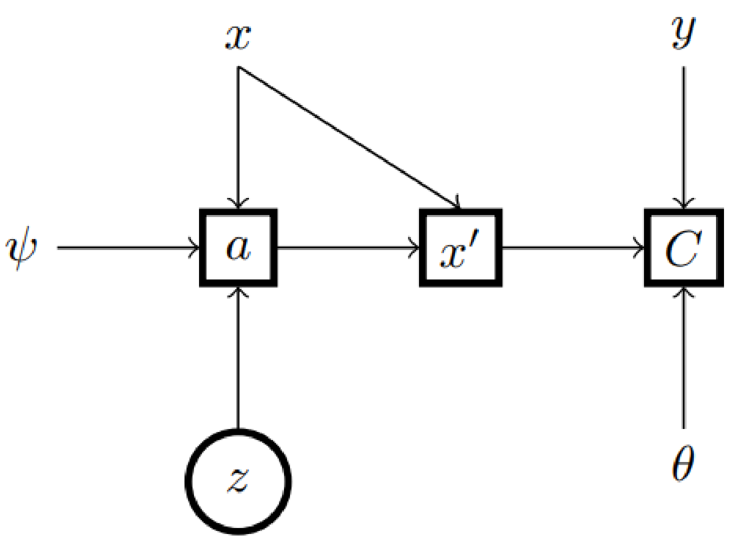
- Square nodes are deterministic and round ones are stochastic
- Inputs and parameters are represented by their vector names

Stochastic Computation Graphs

- The challenge here is that $C(x', y, \theta)$ is non-deterministic and non-differentiable w.r.t the parameters ψ, so we can't use the standard backpropagation algorithm in gradient descent
- Instead we need to somehow estimate $\nabla_{\psi} \mathrm{E}_a[C]$, the gradient of the expected loss w.r.t ψ
  - Two solutions are to use the ***Score Function*** (SF) (Williams 1992) or ***Pathwise Derivative*** (PD) estimator (Schulman et al. 2015)
- Using the ***Score Function*** estimator entails rewriting $\nabla_{\psi} \mathrm{E}_a[C]$ as $\mathrm{E}_a[C \nabla_{\psi} \log(\pi(a|x, \psi))]$ and minimizing a surrogate cost function defined as: $C' = C \log(\pi(a|x, \psi)) + C$
  - This cost function is then regularized to encourage both sparsity in the number of features selected and high variance in the feature subsets selected across examples

Stochastic Computation Graphs

- The *Pathwise Derivative* estimator is also known as the reparametrization trick and was made popular by variational autoencoders (Kingma and Welling 2014)
  - This method entails sampling from $\pi(a|x)$ by first sampling a latent variable z from a known fixed probability distribution $p(z)$ and transforming it, using some function to recover a
  - In variational autoencoders, z is usually sampled from a Gaussian distribution, but the process can be extended to categorical variables by sampling instead from a Gumbel-softmax distribution (Jang, Gu, and Poole 2017; Maddison, Mnih, and Teh 2015)
  - Notice how in the graph a is no longer a stochastic node



Stochastic Computation Graphs

- Assuming K different states, a_{jk} is the probability of assigning state k to feature j :

$$a_{jk} = \frac{exp((log(\hat{q}_{jk}) + g_k)/\tau)}{\sum_{l=1}^{\mathcal{K}} exp((log(\hat{q}_{jl}) + g_l)/\tau)} \text{ for k in } 1, ..., \mathcal{K},$$

  where $g_0, g_1, \ldots, g_k$ are samples from Gumbel(0,1) and τ is a hyperparameter that controls how close the distribution is to the argmax function

  - During training time, $\tau > 0$ ensures that this function is differentiable
    - τ Is gradually decreased as the model reaches convergence
  - At test time, we set $\tau = 0$ to recover the argmax function so that a is once again an indicator variable
  - Now $a \in \mathbb{R}^{d \times K}$, so we must recover the mask before multiplying it by x
    - We do so multiplying a by $w \in \mathbb{R}^K$ so that $aw \in [0,1]^d$ ($w = [0,1]^T$ for $K = 2$)
- PD does not require a surrogate cost function or regularization like SD does, but it does introduce the τ hyperparameter

Experimental Results

Experiment details:

- Models were developed using Tensorflow and run on a GPU
- Refer to the paper for model parameter information

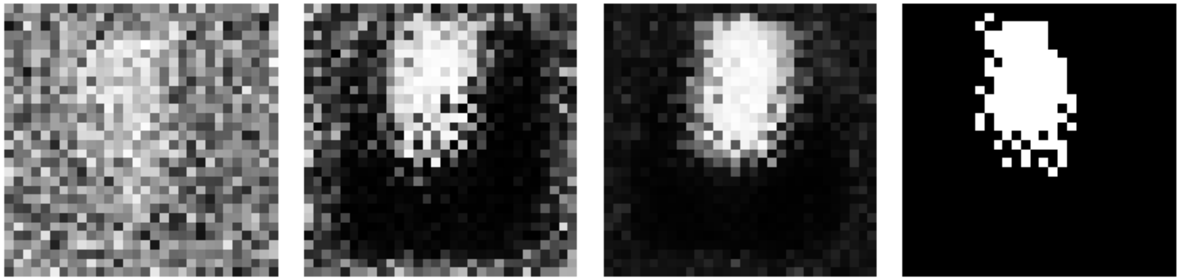
Image Classification Task (proof of concept):

- They used an augmented MNIST handwritten digit dataset
  - Source code and details: github.com/AlCorreia/Human-in-the-loop-Feature-Selection
- In this experiment, user feedback is simulated:

(a) (b)

Experimental Results

- A baseline NN model achieves 92.23% accuracy in the test
- Both SF and PD estimators achieve similar accuracy (88% to 92% after feedback is introduced)
- The figure below shows how the feature selection evolves:
  - Increased sharpness of $\hat{q}$ in (d) shows the effect of the feedback



(a) Beginning (b) 25 epochs (c) 50 epochs (d) feedback

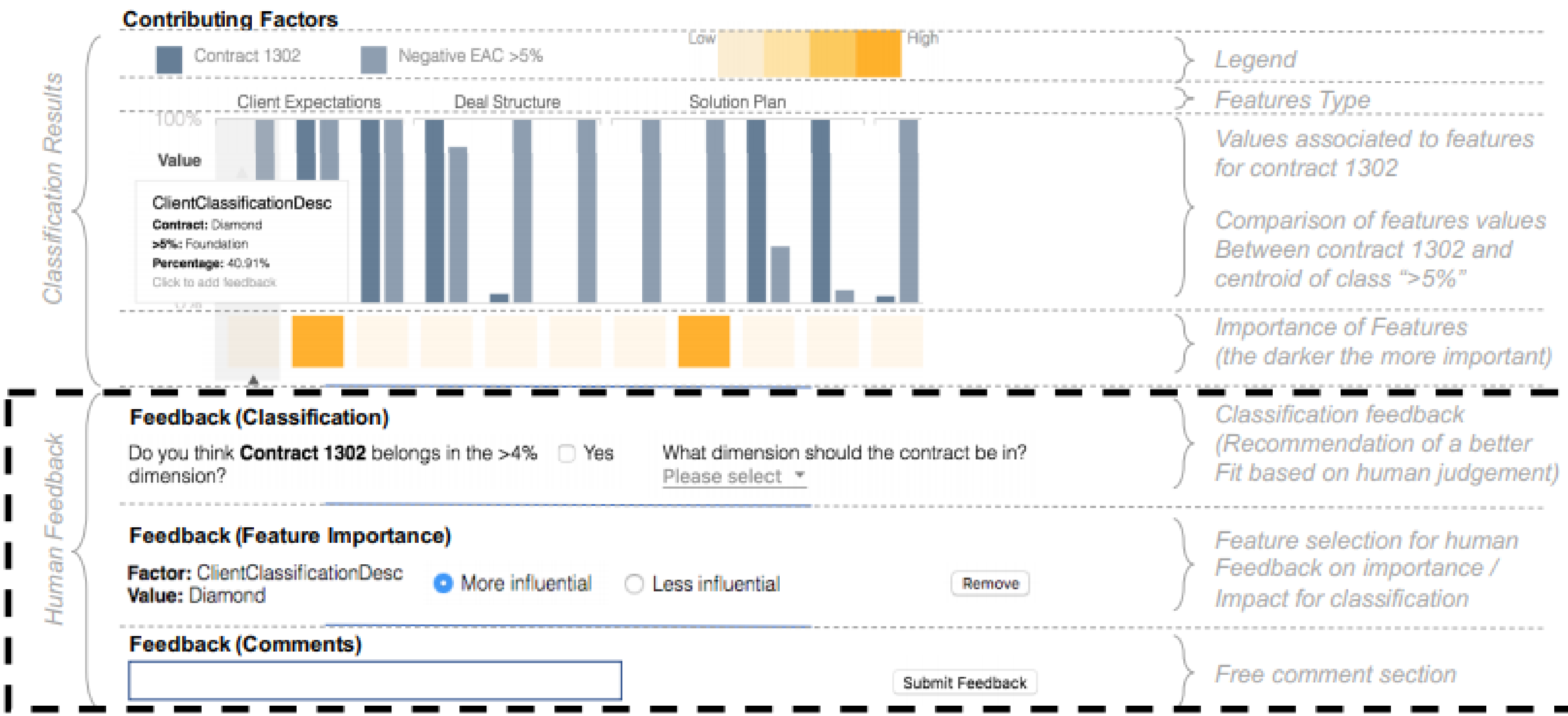
- For PD, a temperature value $\tau = 1.0$ achieved the best results
  - This value presumably balances policy exploration vs feedback exploitation

Experimental Results

Project Risk Classification (PRC) Task:

- The method was also tested on 4 years of project risk profiles
  - The data consists of 349,324 projects across 97 features and is classified among 5 categories, from high to no financial risk)
  - They collected 613,916 pieces of feedback (labels, important features, and comments) on 87,657 active contracts from 114 business experts over a period of 6 months
- This is an interesting test of the model for various reasons:
  - Baseline approaches reach a (low) max accuracy of 31.45% (random forest)
  - PRC is a highly human-curated task that relies on feedback from human experts
  - Completely data-driven approaches would require very large amounts of data

Experimental Results



The user interface for PRC (human feedback in dashed zone)

Experimental Results

- Both SF and PD estimators show significant improvements:

Feedback / Estimator	SF	PD
Before Feedback	29.53	**29.99**
Cosine Feedback	**82.49**	77.51
MSE Feedback	80.11	78.44

- After training with feedback, the models selected a relatively low number of features: 2 to 12 with $\mu: 4.6, \sigma: 1.5$
- Additional observations:
  - The SF estimator does not respond well to dropout, while PD does
  - The estimator used is independent of the architecture, so multiple estimators can be tested and the best for the application can be used
  - The influence of the dissimilarity distance (MSE/cosine) did not affect much, but might vary with the application

Conclusions

- The authors addressed the problem of human-in-the-loop per-example feature selection as a stochastic computation graph.
- They demonstrated that the model could identify the most relevant features of each example in the image classification dataset.
- With the PRC dataset, they showed that the model utilized human feedback to dramatically improve accuracy (31.15% to 81.87%) while also providing business -driven insights.

References

Avdiyenko, L.; Bertschinger, N.; and Jost, J. 2012. Adaptive Sequential Feature Selection for Pattern Classification. In IJCCI International Joint Conference of Computational Intelligence, 474 –482.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine ranslation By Jointly Learning To Align and Translate. International Conference on Learning Representations 2015.

Bengio, Y.; Leonard, N.; and Courville, A. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv:1308.3432 1–12.

Guyon, I., and Elisseeff, A. 2003. An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3:1157–1182.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In International Conference on Learning Representations, 1–13.

Kingma, D. P.; Salimans, T.; Jozefowicz, R.; Chen, X.; Sutskever, I.; and Welling, M. 2016. Improving Variational Inference with Inverse Autoregressive Flow. In Conference on Neural Information Processing Systems, number Nips.

Kohavi, R., and John, G. H. 1997. Wrappers for feature subset selection. Artificial Intelligence 1-2(97):273–324.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2015. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In Advances in Neural Information Processing Systems, 3528—-3536.

Schulman, J.; Heess, N.; Weber, T.; and Abbeel, P. 2015. Gradient Estimation Using Stochastic Computation Graphs. In Advances in Neural Information Processing Systems, 3528—-3536.

Tyagi, V., and Mishra, A. 2013. A Survey on Different Feature Selection Methods for Microarray Data Analysis. International Journal of Computer Applications 67(16):975–8887.

Williams, R. J. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. In Machine learning, number 8, 229–256.

Xu, K.; Courville, A.; Zemel, R. S.; and Bengio, Y. 2015. Show,Attend and Tell : Neural Image Caption Generation with Visual Attention. In International Conference on Machine Learning, 2048–2057.