CPSC 540 Assignment 4 (due March 15 at midnight)

The assignment instructions are the same as for the previous assignment.

1. Name(s): Dingqing Yang, Junjie Zhu
2. Student ID(s): 38800141, 30921167

1 Discrete Markov Chains

1.1 Sampling, Inference, and Decoding

The function *example_markovChain.jl* loads the initial state probabilities and transition probabilities for a Markov chain model,

$$p(x_1, x_2, \dots, x_d) = p(x_1) \prod_{j=2}^{d} p(x_j \mid x_{j-1}),$$

corresponding to the "grad student Markov chain" from class.

1. Write a function, *sampleAncestral*, that uses ancestral sampling to sample a sequence x from this Markov chain of length d. Hand in this code and report the univariate marginal probabilities for time 50 using a Monte Carlo estimate based on 10000 samples. (You can use *sampleDiscrete* in *misc.jl* to sample from a discrete probability mass function).

   Answer: MC estimate of univariate marginal at time 50 for state 1 to 7 is:

   0.01358, 0.38568, 0.01726, 0.00552, 0.10562, 0.08212, 0.39022 respectively

```
function sampleAncestral(p1, pt, d)

    x = zeros(Int8, d)
    # at time 1
    x[1] = sampleDiscrete(p1)
    for i in 2:d
        x[i] = sampleDiscrete(pt[x[i-1], :])
    end
    return x
end
```

```
# use MC estimate to generate 50000 samples
include("sampleAncestral.jl")
acc = zeros(k)
for i = 1:50000
    sample = sampleAncestral(p1, pt, 50)[50]
    acc[sample] += 1
end

marginal_MC = acc / 50000
for i = 1:k
    @printf("marginal of state %d at time 50 using MC is %f\n", i, marginal_MC[i])
end
```

2. Write a function, *marginalCK*, that uses the CK equations to compute the exact univariate marginals up to a given time d. Hand in this code, report all exact univariate marginals, and report how this differs from the marginals in the previous question.

   Answer: Exact marginal at time 50 for state 1 to 7 is:

   0.01393, 0.3863, 0.01702, 0.005424, 0.1041, 0.0843, 0.3889 respectively

```julia
function marginalCK(p1, pt, d, k)
    M = zeros(k, d)
    M[:, 1] = p1
    for i = 2:d
        M[:, i] = pt' * M[:, i-1]
    end
    return M
end
```

```julia
include("marginalCK.jl")
exact_marginal = marginalCK(p1, pt, 50, k)

for i = 1:k
    @printf("exact marginal of state %d at time 50 is %f\n", i, exact_marginal[i, 50])
end
```

3. Write a function, *viterbiDecode*, that uses the Viterbi decoding algorithm for Markov chains to find the optimal decoding up to a time d. Hand in this code and report the optimal decoding of the Markov chain up to time 50 and up to 100, and say how this differs from the sequence defined by the most likely states at each time up to time 100 (the states maximizing the marginal probabilities).

   Answer: Optimum decoding up to time 50 is:
   22222222222222222222222222222222222222222222222222

   Optimum decoding up to time 100 is:
   27777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777777

```julia
function viterbiDecode(p1, pt, d, k)
    #construct M table
    M = zeros(k, d)
    B = zeros(Int8, k, d)

    M[:, 1] = p1
    for i in 2:d
        for c in 1:k
            joint = M[:, i-1] .* pt[:, c]
            M[c, i] = maximum(joint)
            B[c, i] = argmax(joint)
        end
    end

    # return decoding
    decoded = zeros(Int8, d)
    decoded[d] = argmax(M[:, d])
    for i in d:-1:2
        decoded[i-1] = B[decoded[i], i]
    end
    return decoded
end
```

```
include("viterbiDecode.jl")
time = 50
println("time is: ", time)
decoded = viterbiDecode(p1, pt, time, k)

println("Viterbi Decoded sequences: ")
for i = 1:time
    print(decoded[i])
end
println("")

time = 100
println("time is: ", time)
decoded = viterbiDecode(p1, pt, time, k)

println("Viterbi Decoded sequences: ")
for i = 1:time
    print(decoded[i])
end
println("")
```

State maximizing the marginal up to time 100 is:

22222222222222222222222222222222222222222222227777777777777777777777777777777777777777777777777777

```
# state sequences by maximizing marginal
println("state sequences obtained by maximizing marginal")
M = marginalCK(p1, pt, time, k)
for i = 1:time
    print(argmax(M[:, i]))
end
println("")

```

4. Report all the univariate conditional probabilities at time 50 if the student starts in grad school, $p(x_{50} = c \mid x_1 = 3)$ for all c.

   Answer: Univariate conditional at time 50 for state 1 to 7 is:

   0.008336, 0.2311, 0.01019, 0.008489, 0.1831, 0.1699, 0.3889 respectively

```
init = zeros(Int8, k)
init[3] = 1 # start for grad school
conditional = marginalCK(init, pt, 50, k)
for i = 1:k
    @printf("exact marginal of state %d at time 50 starting from grad school is %f\n", i, conditional[i, 50])
end
```

Hint: for parts 2-3, you can use a 7 by d matrix M to represent the dynamic programming table, and for part 3 you can use another matrix B containing the argmax values that lead to each entry in the table. For the conditional question, you can answer it by changing the input to the existing code.

1.2 Conditioning Queries Requiring Inference

Next consider the following cases (which require implementing an extra rejection step or backward phase):

1. Report all the univariate conditional probabilities $p(x_5 = c \mid x_{10} = 6)$ ("where are you likely to be 5 years after graduation if you are in academia 10 years") obtained using a Monte Carlo estimate based on 10000 samples and rejection sampling. Also report the number of samples accepted among the 10000 samples.

Answer: MC estimate of $p(x_5 = c \mid x_{10} = 6)$ using rejection sampling for $x_5 = 1$ to 7 is:

0.0000, 0.03563, 0.2779, 0.05938, 0.1045, 0.5226, 0.0000 respectively

Number of accepted samples = 421

```
# use MC estimate to generate 50000 samples
include("sampleAncestral.jl")
joint_count = zeros(k)
accepted = 0
for i = 1:10000
    sample = sampleAncestral(p1, pt, 10)
    if sample[10] == 6
        global accepted += 1
        joint_count[sample[5]] += 1
    end
end
marginal_cond = joint_count ./ accepted
@printf("Number of accepted samples: %d\n", accepted)
for i = 1:k
    @printf("MCMC conditional marginal of state %d is %f\n", i, marginal_cond[i])
end
```

2. Write a function, *sampleBackwards* that uses backwards sampling to sample sequences of length d given a particular value of the last value x_d. Hand in this code and report all the univariate conditional probabilities $p(x_5 = c \mid x_{10} = 6)$ obtained using a Monte Carlo estimate based on 10000 samples.

   Answer: MC estimate of $p(x_5 = c \mid x_{10} = 6)$ using backwards sampling for $x_5 = 1$ to 7 is:

   0.0013, 0.0361, 0.2453, 0.0542, 0.1714, 0.4917, 0.0000 respectively

```
include("marginalCK.jl")
margin = marginalCK(p1, pt, 10, k)
# construct reverse transition matrix
pr = zeros(10-5, k, k)
for time = 1:5
    for from = 1:k
        for to = 1:k
            pr[time, from, to] = pt[to, from] * margin[to, 10-time] / margin[from, 11-time]
        end
    end
end

# print(sum(pr, dims=3)) # sanity check

acc = zeros(k) # accumulator for counting
for i = 1:10000
init = zeros(Int8, k)
init[6] = 1 # start for grad school
    for j = 1:5
        global bk_sample = sampleAncestral(init, pr[j, :, :], 2)[2]
        # reset init
        init = zeros(Int8, k)
        init[bk_sample] = 1
        # print(typeof(sample))
    end
    # println(bk_sample)
    acc[bk_sample] += 1
end

backward_cond = acc ./ 10000.0
for i = 1:k
    @printf("MCMC conditional marginal of state %d is %f\n", i, backward_cond[i])
end
```

3. Write a function, *forwardBackwards* that is able compute all exact univariate conditionals $p(x_j \mid x_d = c)$ in $O(dk^2)$ for a length-d sentence. Hand in the code and report all the exact univariate conditionals $p(x_5 = c \mid x_{10} = 6)$.

   Answer: Exact $p(x_5 = c \mid x_{10} = 6)$ using forwardbackwards algorithm for $x_5 = 1$ to 7 is:

0.001554, 0.03393, 0.2477, 0.05807, 0.1613, 0.4974, 0.0000 respectively

```
function marginalBackward(pd, pt, d, k)
    #
    V = zeros(k, d)
    V[:, d] = pd
    for i = d:-1:2
        V[:, i-1] = pt * V[:, i]
    end
    return V
end
```

```
# compute prob(x_j | x_d = c)
function forwardBackwards(p1, pt, j, d, c, k)
    #forward
    M = marginalCK(p1, pt, d, k)

    # backward
    pd = zeros(k)
    pd[c] = 1
    V = marginalBackward(pd, pt, d, k)
    result = M[:, j] .* V[:, j]
    return 1 ./ sum(result) * result
end
```

```
# exact univariate conditional
uni_cond = forwardBackwards(p1, pt, 5, 10, 6, k)
for i = 1:k
    @printf("univariate conditionals of state %d is %f\n", i, uni_cond[i])
end
```

2 Directed Acyclic Graphical Models

2.1 D-Separation

Consider a directed acyclic graphical (DAG) model with the following graph structure:

Assuming that the conditional independence properties are faithful to the graph, using d-separation briefly explain why the following are true or false:

1. $H \perp I$.

   Answer: False: Path IAH not blocked

2. $H \perp I \mid A$.

   Answer: True: Path IAH has a fork at A, A observed. Path IJH: v structure: common child J not observed.

3. $H \perp I \mid J$.

   Answer: False: Path IJH: v structure, common child J observed.

4. $H \perp I \mid A, J$.

   Answer: False: Path IJH: v structure, common child J observed.

5. $C \perp I$.

Answer: True: Path IADHEC, DE has v-structure with descendants not observed. Path IJHEC, IH has v-structure with descendants not observed.

6. $C \perp I \mid J$.

   Answer: False: Path IJHEC, IH has v-structure with child J observed. No other way of blocking this path.

7. $C \perp I \mid H$.

   Answer: False: Path IADHEC, DE has v-structure with child H observed. No other way to block this path.

8. $C \perp I \mid A, H$.

   Answer: True: Path IADHEC, A is a fork and A is observed. Path IJHEC, H is the middle of a chain and it is observed.

9. $C \perp I \mid E, H, J$.

   Answer: True: Both path IADHEC and IJHEC are blocked by E in the middle of a chain.

2.2 Exact Inference

Consider a directed acyclic graphical (DAG) model with the following graph structure: Assume that all variables are binary and that we use the following parameterization of the network:

$$
\begin{aligned}
&p(A = 1) = 0.7 \\
&p(B = 1 \mid A = 0) = 0.8 \\
&p(B = 1 \mid A = 1) = 1.0 \\
&p(C = 1) = 0.8 \\
&p(D = 1 \mid B = 0) = 0.8 \\
&p(D = 1 \mid B = 1) = 0.6 \\
&p(E = 1 \mid B = 0, C = 0) = 0.3 \\
&p(E = 1 \mid B = 0, C = 1) = 0.7 \\
&p(E = 1 \mid B = 1, C = 0) = 0.4 \\
&p(E = 1 \mid B = 1, C = 1) = 0.5 \\
&p(F = 1 \mid A = 0) = 0.5 \\
&p(F = 1 \mid A = 1) = 0.9 \\
&p(G = 1 \mid E = 0, F = 0) = 0.5 \\
&p(G = 1 \mid E = 0, F = 1) = 0 \\
&p(G = 1 \mid E = 1, F = 0) = 0.1 \\
&p(G = 1 \mid E = 1, F = 1) = 0.1
\end{aligned}
$$

Answer:

1. $p(A = 0) = 1 - p(A = 1) = 0.3$.
2. $p(B = 1 \mid A = 0) = 0.8$.
3. $p(B = 1) = p(B = 1 \mid A = 0)p(A = 0) + p(B = 1 \mid A = 1)p(A = 1) = 0.8 \times 0.3 + 1 \times 0.7 = 0.94$.
4. $p(D = 1) = p(D = 1 \mid B = 0)p(B = 0) + p(D = 1 \mid B - 1)p(B = 1) = 0.8 \times (1 - 0.94) + 0.6 * 0.94 = 0.612$.

5. $p(B = 1 \mid D = 1) = \frac{p(D=1 \mid B=1)p(B=1)}{p(D=1)} = \frac{0.6 \times 0.94}{0.612} = 0.62745.$
6. $p(B = 1 \mid C = 1) = p(B = 1) = 0.94.$
7. $p(B = 1 \mid A = 0, C = 1, F = 1) = p(B = 1 \mid A = 0) = 0.8.$

Hints: some of the above quantities can be read from the table, some require using that probabilities sum to 1, some require the marginalization rule, some require Bayes rule, some require using [conditional] independence, and some will be simplified using calculations from previous sub-questions.

2.3 Inpainting

The function *example_fil.jl* loads a variant of the MNIST dataset. It contains all the training images but the test images are missing their bottom half. Running this function fits an independent Bernoulli model to the training set, and then shows the result of applying the density model to "fill in" four random test examples. It performs pretty badly because the independent model can't condition on the known top-half of the images.

1. Make a variant of the demo where you fit an inhomogeneous Markov chain to each image column. Hand in your code and an example of using this model to fill in 4 random test images.

   Answer:

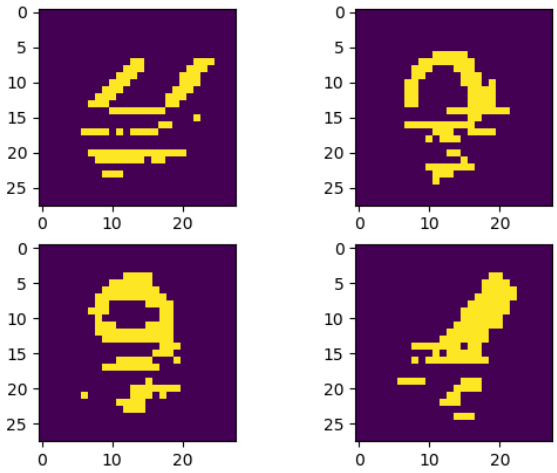
```
include("inhomogMarkov.jl")
subModel = Array{SampleModel}(undef,m,m)

# Train an inhomog Markov chain
for i in 1:m
    for j in 1:m
        if j != 1
            subModel[i,j] = inhomogMarkov(X[i,j-1,:], X[i,j,:])
        else
            subModel[i,1] = inhomogMarkov(zeros(n), X[i,1,:])
        end
    end
end
```

   Where inhomogMarkov function is modified from *tabular.jl* with the same sample function (omitted in screenshot for clarity)

```
function inhomogMarkov(prev,curr)
    n = size(prev,1)

    # Compute the frequencies in the training data
    # (The below is *not* efficient in time or space)
    D =  Dict()
    for i in 1:n
        key = [curr[i];prev[i]]
        if haskey(D,key)
            D[key] += 1
        else
            D[key] = 1
        end
    end
```



2. Make a variant of the demo where you fit a directed acyclic graphical model to the data, using general discrete conditional probabilities and where the parents of pixel (i, j) are the other 8 pixels in the region $(i-2:i, j-2:j)$. Hand in your code and an example of using this model to fill in 4 random test images.

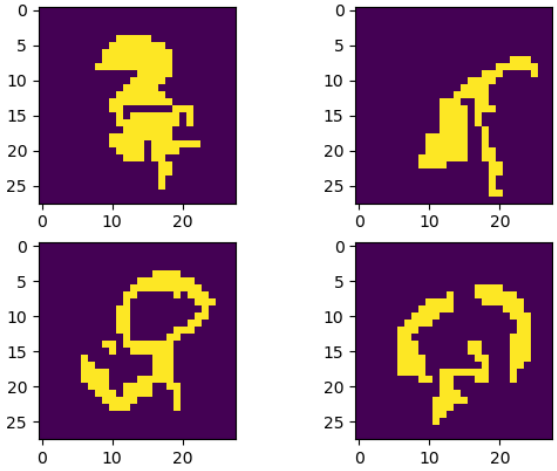
   Answer: No padding is performed here since the edge of the picture is inactive pixels.

```
# Train a DAG model using tabular approach
for i in 3:m
    for j in 3:m
        subModel[i,j] = DAG(X[i-2:i,j-2:j,:])
    end
end
```

   Where DAG function is modified from *tabular.jl* with the same sample function (omitted in screenshot for clarity)

```
function DAG(tile) # 3 by 3 pixel containing parent and child
    n = size(tile,3)

    # Compute the frequencies in the training data
    # (The below is *not* efficient in time or space)
    D =  Dict()
    for i in 1:n
        key = tile[:, :, i][:]
        if haskey(D,key)
            D[key] += 1
        else
            D[key] = 1
        end
    end
```

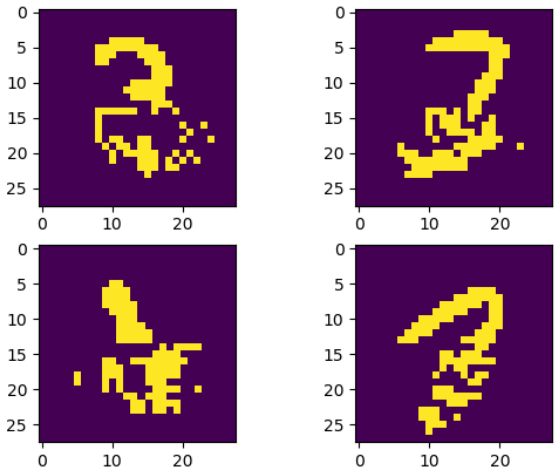


3. Make a variant of the demo where you fit a sigmoid belief network to the data, where the parents of pixel (i, j) are the other pixels in the region $(1 : i, 1 : j)$. Hand in your code and an example of using this model to fill in 4 random test images.

Answer: No padding performed for the same reason above. SBN directly trained using *logreg.jl* provided

```
include("logReg.jl")
subModel = Array{SampleModel}(undef,m,m)

# Train a sigmoid belief network
for i in 2:m
    for j in 2:m
        Xtrain = reshape(X[1:i, 1:j, :], (:, n))
        subModel[i,j] = logReg(Xtrain[1:end-1, :]', Xtrain[end, :])
    end
end
```



Hint: you may find it helpful to make an m by m array called *models* where element (i, j) contains the model

for pixel (i, j). Included in *a4.zip* are *tabular.jl* and *logreg.jl* which implement solving a supervised learning problem using the tabular and sigmoid method (respectively). You may want to debug on a smaller version of the training set, since the runtime of fitting these models is non-trivial (solving 784 supervised learning problems takes time, although you could parallelize to make this go very quickly).

3 Very-Short Answer Questions

Give a short and concise 1-sentence answer to the below questions.

1. Describe a procedure to sample from the (continuous) exponential distribution.

   Answer: (L19) First, sample u from the uniform distribution on [0,1]. Then use the inverse CDF F^{-1} for exponential distribution to compute $F^{-1}(u)$.

2. Describe a procedure to sample from the (discrete) Poisson distribution.

   Answer: (L19) First, sample u from the uniform distribution on [0,1]. Then use the inverse CDF F^{-1} for the discrete Poission distribution to compute $F^{-1}(u)$.

3. What is an advantage of using Monte Carlo methods over the CK equations to estimate probabilities in Markov chains?

   Answer: (L20) Runtime for CK is $O(dk^2)$, where generating one sample from Monte Carlo takes $O(d \log k)$, which is better if the number of samples that we generate is less that k^2

4. Why don't we normally consider the stationary distribution of inhomogeneous Markov chains?

   Answer: In general, an inhomogeneous Markov chain is not stable, so the stationary distribution does not exist. Reference: https://mathoverflow.net/questions/244685/convergence-of-an-inhomogeneous-markov-chain

5. Suppose we are using a hidden Markov model to track the location of a submarine using sonar measurements. What would x_j and z_j represent in this example?

   Answer: z_j: whether the submarine is at the surface of the sea or not. x_j: distances

6. Given a Markov model over x with discrete x_j, describe the parameterization of a hidden Markov models that defines the same distribution over x.

   Answer: $x_j = z_j$, or $P(x_j = z_j | z_j) = 1$

7. What is an advantage and a disavantage of using logistic regression to parameterize the CPDs in DAGs compared to a tabular representation?

   Answer: (L23) Advantage: Don't need too many parameters (one parameter is needed for each combination of the x_i
   Disadvantage: Fitting many independent models is very time comsuming if no parallel hardware available.

8. When decoding a DAG, why does the order that we compute the messages matter?

   Answer: (L24) interference and messsage passing will be exponentially more expensive in terms of runtime with the wrong ordering.

9. What is the relationship between Ising models and UGMs?

   Answer: (L24) Ising is a special case of UGM, where $\phi_i(x_i) = \exp(x_i w_i)$, $\phi_{ij}(x_i, x_j) = \exp(x_i x_j w_i j)$.

10. What is the relevance of the Markov blanket in ICM?

    Answer: (L25) They exclude other nodes that are not needed when one variable's probability is computed and maximized.

11. Why do we have a "burn in" phase for Gibbs sampling?

    Answer: (L25) We need to throw away early samples when we are not close to being stationary since they are very likely to be quite off from what the model actually describes.

4 Relevant Papers for Project

4.1 Finding Relevant Papers

To help you make progress on your project, for this part you should hand in a list of 10 academic papers related to your current project topic. Finding related work is often one of the first steps towards getting a new project started, and it gives you an idea of what has (and has not) been explored. Some strategies for finding related papers are:

1. Use Google: try the keywords you think are most relevant. Asking people in your lab (or related labs) for references is also often a good starting point.
2. Once you have found a few related papers, read their introduction section to find references that these papers think are worth mentioning.
3. Once you have found a few related papers, use Google Scholar to look through the list of references that are *citing* these papers (particularly for recent ones). You may have to do some sifting if there are a lot of citations. Reasonable criteria to sift through large reference lists include looking for the ones with the most citations or focusing on the more recent ones (then returning to Step 2 to find the more-relevant older references).

For this question you only need to provide a list, but in Assignment 5 you will have to do a survey of 10 papers. So it's worth trying to identify papers that are both relevant and important at this point. For some types of projects it will be easier to find papers than others. If you are having trouble, post on Piazza.

Although the papers do not need to all be machine learning papers, the course project does need to be related to machine learning in some way, so at least a subset of the papers should be machine learning papers. Here is a rough guide to some of the most reputable places to where you see machine learning works published:

- The International Conference on Machine Learning (ICML) and the conference on Advances in Neural Information Processing (NeurIPS) are the top places to publish machine learning work. The Journal of Machine Learning Research (JMLR) is the top journal, although in this field conference publications are usually viewed as more prestigious.
- Other good venues include AISTATS (emphasis on statistics), UAI (emphasis on graphical models), COLT (emphasis on theory), ICLR (emphasis on deep learning), ECML-PKDD (European version of ICML), CVPR and ICCV/ECCV (emphasis on computer vision), ACL and EMNLP (emphasis on language), KDD (emphasis on data mining), AAAI/IJCAI (emphasis on AI more broadly), JRSSB and Annals of Stats (emphasis on statistics more broadly), and Science and Nature (emphasis on science more broadly).

Answer:

1. Neural architecture search with reinforcement learning. [ICLR 2017]

2. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. [IJCAI 2015]
3. Network Morphism [ICML 2016]
4. Efficient Neural Architecture Search via Parameter Sharing. [ICML 2018]
5. Understanding and Simplifying One-Shot Architecture Search. [ICML 2018]
6. Progressive Neural Architecture Search. [ECCV 2018]
7. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. [ICML 2018]
8. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. [ECCV 2018]
9. meProp: Sparsified Back Propagation for Accelerated Deep Learning with Reduced Overfitting. [ICML 2017]
10. Full deep neural network training on a pruned weight budget. [SYSML 2019]

4.2 Paper Review

Among your list of 10 papers, choose one paper and write a review of this paper. It makes sense to choose a paper that is closely-related to your project or to choose one of the most important-looking papers. The review should have two parts:

1. A short summary of the contributions of the paper. Say what problem the paper is addressing, why this is an important problems, what is proposed, and how it is being evaluated.
2. A list of strengths and weaknesses of the paper, and whether the claims are appropriately evaluated. For ideas of what issues to discuss, see the JMLR guidelines for reviewers:
   `http://www.jmlr.org/reviewer-guide.html`

Note that you should include a non-empty list of strengths *and* weaknesses. Many students when doing their first reviews focus either purely on strengths or purely on weaknesses. It's important to recognize that all papers have weaknesses or limitations (even ones written by famous people or that are published in impressive places or that proved to be historically important) and all papers have strengths or at least a motivation (the authors must have thought it was worth writing for some reason).

Answer: We choose to write the review for Neural architecture search with reinforcement learning. [ICLR 2017]. Paper review is concatenated in the next page.

Paper Review: Neural Architecture Search with Reinforcement Learning [Zoph and Le, 2016]

1 Contribution

The success of deep learning has shifted the focus of field from feature engineering to architecture engineering. Zoph and Le [2016] acknowledge this and believe that it is critical to reduce required expert knowledge and human labour when designing neural networks. The authors formulate architecture hyperparameters (e.g. size of filters within a convolution layer) as a variable length sequence generated by a recurrent network (i.e. a controller). The resulting network (i.e. a child network) is then trained and evaluated to obtain the validation accuracy. The authors use it as the reward signal and evaluate policy gradient to update the controller parameter. Therefore, the controller is likely to generate better architecture over time. This process is summarized in Figure 1. Evaluation is performed on both vision and language task. Result that is similar to previous state-of-the-art is achieved on CIFAR-10 dataset, while new state-of-the-art result is achieved on Penn Treebank.

2 Strength

- *This work shows state-of-the-art results on a language task that beats human-designed models.* Their model achieves 3.6 perplexity better than previous state-of-the-art.

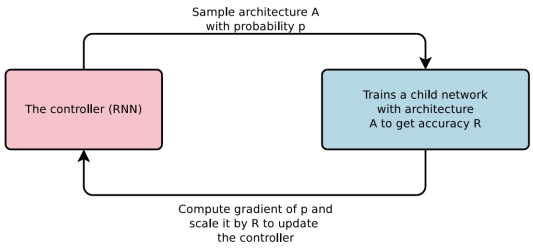


Figure 1: Nueral Architecture Search Overview

- *The proposed method works in multiple domains.* While most previous NAS methods focus on vision task, this is the first work that apply NAS on language modeling [Elsken et al., 2018]. [Zoph and Le, 2016] achieve both state-of-the-art in representative vision and language benchmarks.
- *Detailed experimental results and comparisons are provided.* The authors evaluate and compare many state-of-the-art models with their own model in terms of test accuracy and model sizes.

3 Weakness

- *The proposed method is extremely computationally expensive.* It takes a few weeks for the authors to search best architecture with 800 GPUs. Thousands of GPU days are too expensive for most individuals or academic institutes. Many possible ways to reduce the cost include learning curve extrapolation [Domhan et al., 2015], network morphism [Wei et al., 2016], and one-shot NAS [Pham et al., 2018].
- *Search space is limited to architecture hyperparameters only instead of all hyperparameters.* Other hyperparameters such as learning rate are not searched, and the authors use a fixed scheme for all found architectures.
- *The use of human prior limits the novelty of searched models.* The search choices only include common operations like convolution with different hyperparamters (e.g. kernel size and stride) and skip connection. It cannot propose true novel architecture but a complicated composition of existing components.
- *Limited cell interpretability.* As mentioned in the previous point, it only proposes some existing components with complex interaction, which is more difficult to interpret. Also, it lacks the ability to reason about why certain generated architectures work well or not in a particular setting.

References

Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *International Conference on Machine Learning*, pages 564–572, 2016.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.