CZSL project overview

February 28, 2020

What was done in October

1. Continual learning pipeline for CUB/AwA
2. Sequential/EWC/MAS/A-GEM baselines for classification with attributes on CUB/AwA datasets
3. Some metrics to measure performance for continual zero-shot learning

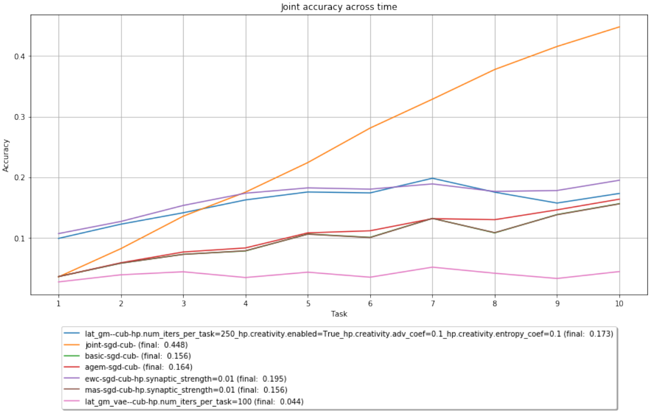


Figure: Joint accuracy on all the remaining unseen classes for CUB

What was done in November

- LGM-GAN on CUB dataset with the *pretrained and fixed* embedder
- LGM-VAE on CUB dataset with the *pretrained and fixed* embedder
  - with/without the learned prior
  - with/without class attributes in VAE/in Classifier
- CL “validation” pipeline from A-GEM paper, trying different optimizers, trying model reinits, tweaking architectures, moving workflow into slurm, some metrics (LCA, forgetting speed) from A-GEM paper, fixing several bugs

Why we have dropped MeRGAN

Why we have dropped MeRGAN

Fact: original MeRGAN operates in the image space.

Why we have dropped MeRGAN

Fact: original MeRGAN operates in the image space.
Claim: applying the creativity loss in the image space is not good.

Why we have dropped MeRGAN

Fact: original MeRGAN operates in the image space.
Claim: applying the creativity loss in the image space is not good.
Justification:

- In CIZSL, the creativity loss was applied to features, not images, i.e. we didn't have a good precedent of using the creativity loss in the image space to improve model's performance on unseen.
- It feels hard to manipulate images to generate sensible unseen images
- We would need a very big GAN model to do that, which is slower to train and harder to tune

Why we have dropped MeRGAN

Fact: original MeRGAN operates in the image space.
Claim: applying the creativity loss in the image space is not good.
Justification:

- In CIZSL, the creativity loss was applied to features, not images, i.e. we didn't have a good precedent of using the creativity loss in the image space to improve model's performance on unseen.
- It feels hard to manipulate images to generate sensible unseen images
- We would need a very big GAN model to do that, which is slower to train and harder to tune

Idea #1: let's apply creativity in the feature space, sticking closer to CIZSL
Problem: where can we get these features?

Why we have dropped MeRGAN

Fact: original MeRGAN operates in the image space.
Claim: applying the creativity loss in the image space is not good.
Justification:

- In CIZSL, the creativity loss was applied to features, not images, i.e. we didn't have a good precedent of using the creativity loss in the image space to improve model's performance on unseen.
- It feels hard to manipulate images to generate sensible unseen images
- We would need a very big GAN model to do that, which is slower to train and harder to tune

Idea #1: let's apply creativity in the feature space, sticking closer to CIZSL
Problem: where can we get these features?
Idea #2: let's just use features from Classifier

Using features from Classifier (1/3)

Using features from Classifier (1/3)

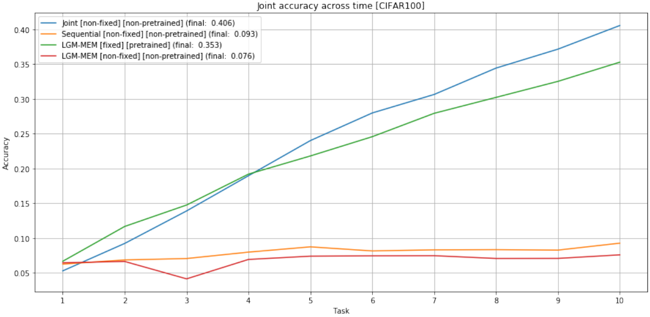
Claim It doesn't work as is.

Using features from Classifier (1/3)

Claim It doesn't work as is.
Intuition. Features drift away and LGM produces old features that are not only useless but even detrimental.

Using features from Classifier (1/3)

Claim It doesn't work as is.
Intuition. Features drift away and LGM produces old features that are not only useless but even detrimental.
Justification



Using features from Classifier (2/3)

Using features from Classifier (2/3)

Idea #3: what if we'll use a pretrained feature extractor?

Using features from Classifier (2/3)

Idea #3: what if we'll use a pretrained feature extractor?

Claim: it has 2 serious problems:

Using features from Classifier (2/3)

Idea #3: what if we'll use a pretrained feature extractor?

Claim: it has 2 serious problems:

- It *feels* unfair to use pretrained models

Using features from Classifier (2/3)

Idea #3: what if we'll use a pretrained feature extractor?

Claim: it has 2 serious problems:

- It *feels* unfair to use pretrained models
- It *feels* not novel (ICCV19 paper does that, for example)

Using features from Classifier (2/3)

Idea #3: what if we'll use a pretrained feature extractor?
Claim: it has 2 serious problems:

- It *feels* unfair to use pretrained models
- It *feels* not novel (ICCV19 paper does that, for example)

Idea #4: ok then, let's apply EWC to the Classifier to prevent features from drifting away.

Using features from Classifier (2/3)

Idea #3: what if we'll use a pretrained feature extractor?
Claim: it has 2 serious problems:

- It *feels* unfair to use pretrained models
- It *feels* not novel (ICCV19 paper does that, for example)

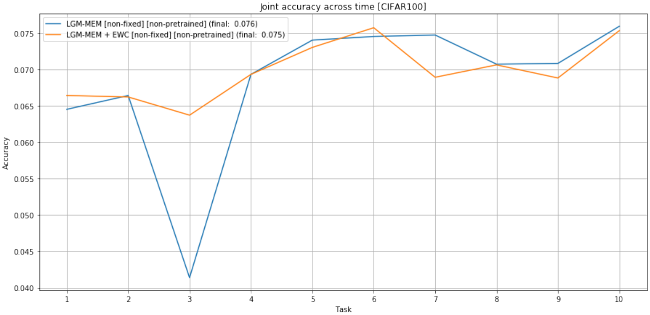
Idea #4: ok then, let's apply EWC to the Classifier to prevent features from drifting away.
Claim: it did not help, but I am sure I have a bug...

Using features from Classifier (2/3)

Idea #3: what if we'll use a pretrained feature extractor?
Claim: it has 2 serious problems:

- It *feels* unfair to use pretrained models
- It *feels* not novel (ICCV19 paper does that, for example)

Idea #4: ok then, let's apply EWC to the Classifier to prevent features from drifting away.
Claim: it did not help, but I am sure I have a bug...
Justification



Using features from Classifier (3/3)

We can go into improving this approach, but let's step back for a while and think about LGM on top of the Classifier's features more generally. It has two serious problems:

Using features from Classifier (3/3)

We can go into improving this approach, but let's step back for a while and think about LGM on top of the Classifier's features more generally. It has two serious problems:

- We can't learn classes one by one like MerGAN can:

Using features from Classifier (3/3)

We can go into improving this approach, but let's step back for a while and think about LGM on top of the Classifier's features more generally. It has two serious problems:

- We can't learn classes one by one like MerGAN can:
  - because of that it will not be a replacement for MeRGAN

Using features from Classifier (3/3)

We can go into improving this approach, but let's step back for a while and think about LGM on top of the Classifier's features more generally. It has two serious problems:

- We can't learn classes one by one like MerGAN can:
  - because of that it will not be a replacement for MeRGAN
  - because of that it will not be possible to use it in modern CL setups without task identity or task bounds

Using features from Classifier (3/3)

We can go into improving this approach, but let's step back for a while and think about LGM on top of the Classifier's features more generally. It has two serious problems:

- We can't learn classes one by one like MerGAN can:
  - because of that it will not be a replacement for MeRGAN
  - because of that it will not be possible to use it in modern CL setups without task identity or task bounds
- Its performance will depend a lot on EWC performance

What was done in December

What was done in December

- Some small features: label smoothing, tuning joint baseline, etc

What was done in December

- Some small features: label smoothing, tuning joint baseline, etc
- Add EWC/MAS regularization to LGM

What was done in December

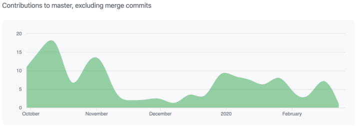
- Some small features: label smoothing, tuning joint baseline, etc
- Add EWC/MAS regularization to LGM
- Some bug fixes (proper GP calculation, metrics, model cloning, etc)

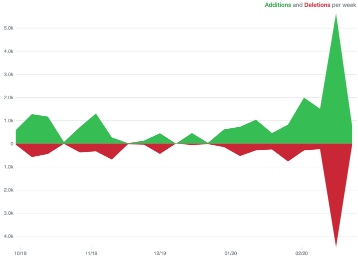
What was done in December

- Some small features: label smoothing, tuning joint baseline, etc
- Add EWC/MAS regularization to LGM
- Some bug fixes (proper GP calculation, metrics, model cloning, etc)
- December was quite chaotic because of the conference, paperwork, holidays, etc...

What was done in December

- Some small features: label smoothing, tuning joint baseline, etc
- Add EWC/MAS regularization to LGM
- Some bug fixes (proper GP calculation, metrics, model cloning, etc)
- December was quite chaotic because of the conference, paperwork, holidays, etc...





What was done in January

What was done in January

- LGM-AC-GAN and LGM-VAE without attributes on CUB
  - Result: 19.1% and 33% FJA.

What was done in January

- LGM-AC-GAN and LGM-VAE without attributes on CUB
  - Result: 19.1% and 33% FJA.
- Convolutional LGM-GAN (like in ICCV19 paper) on CUB
  - Result: performance dropped to 17.4% FJA

What was done in January

- LGM-AC-GAN and LGM-VAE without attributes on CUB
  - Result: 19.1% and 33% FJA.
- Convolutional LGM-GAN (like in ICCV19 paper) on CUB
  - Result: performance dropped to 17.4% FJA
- Creativity loss via simple entropy
  - Result: nothing changed (but I didn't experiment a lot)

What was done in January

- LGM-AC-GAN and LGM-VAE without attributes on CUB
  - Result: 19.1% and 33% FJA.
- Convolutional LGM-GAN (like in ICCV19 paper) on CUB
  - Result: performance dropped to 17.4% FJA
- Creativity loss via simple entropy
  - Result: nothing changed (but I didn't experiment a lot)
- Slow MeRGAN baseline on SVHN and AwA
  - Result: 60% CAS for SVHN

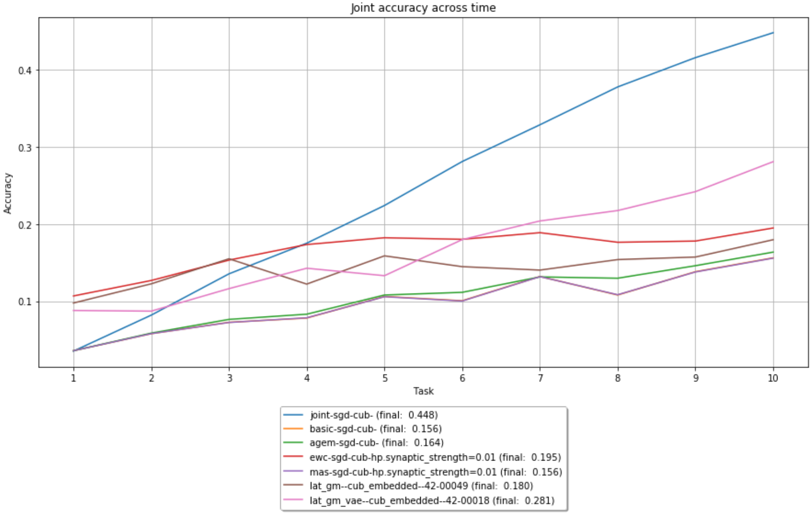
What was done in January

- LGM-AC-GAN and LGM-VAE without attributes on CUB
  - Result: 19.1% and 33% FJA.
- Convolutional LGM-GAN (like in ICCV19 paper) on CUB
  - Result: performance dropped to 17.4% FJA
- Creativity loss via simple entropy
  - Result: nothing changed (but I didn't experiment a lot)
- Slow MeRGAN baseline on SVHN and AwA
  - Result: 60% CAS for SVHN
- Some small things: CL setup from ICCV19 paper (i.e. a large first task and small subsequent tasks), task transfer metric, etc.

What was done in January

- LGM-AC-GAN and LGM-VAE without attributes on CUB
  - Result: 19.1% and 33% FJA.
- Convolutional LGM-GAN (like in ICCV19 paper) on CUB
  - Result: performance dropped to 17.4% FJA
- Creativity loss via simple entropy
  - Result: nothing changed (but I didn't experiment a lot)
- Slow MeRGAN baseline on SVHN and AwA
  - Result: 60% CAS for SVHN
- Some small things: CL setup from ICCV19 paper (i.e. a large first task and small subsequent tasks), task transfer metric, etc.
  - Result: 37% final joint accuracy on CUB (ICCV19 paper has 53%, but they do a lot of "cheating")

LGM-GAN vs LGM-VAE



LGM is cumbersome

LGM is cumbersome

- Several terms (8 in total) in the objective:

LGM is cumbersome

- Several terms (8 in total) in the objective:
  - AC-GAN losses: generator loss, generator classification loss, discriminator loss, classifier loss, gradient penalty

LGM is cumbersome

- Several terms (8 in total) in the objective:
  - AC-GAN losses: generator loss, generator classification loss, discriminator loss, classifier loss, gradient penalty
  - Generator prototypipcal loss

LGM is cumbersome

- Several terms (8 in total) in the objective:
  - AC-GAN losses: generator loss, generator classification loss, discriminator loss, classifier loss, gradient penalty
  - Generator prototypipcal loss
  - Rehearsal losses: generator rehearsal loss, classifier rehearsal loss

LGM is cumbersome

- Several terms (8 in total) in the objective:
  - AC-GAN losses: generator loss, generator classification loss, discriminator loss, classifier loss, gradient penalty
  - Generator prototypipcal loss
  - Rehearsal losses: generator rehearsal loss, classifier rehearsal loss
- Complex training scenario
  - Continual learning involves several stages
  - LGM/Classifier training stages
- A lot of hyperparameters to tune:

LGM is cumbersome

- Several terms (8 in total) in the objective:
  - AC-GAN losses: generator loss, generator classification loss, discriminator loss, classifier loss, gradient penalty
  - Generator prototypipcal loss
  - Rehearsal losses: generator rehearsal loss, classifier rehearsal loss
- Complex training scenario
  - Continual learning involves several stages
  - LGM/Classifier training stages
- A lot of hyperparameters to tune:
  - Optimizers for each component

LGM is cumbersome

- Several terms (8 in total) in the objective:
  - AC-GAN losses: generator loss, generator classification loss, discriminator loss, classifier loss, gradient penalty
  - Generator prototypipcal loss
  - Rehearsal losses: generator rehearsal loss, classifier rehearsal loss
- Complex training scenario
  - Continual learning involves several stages
  - LGM/Classifier training stages
- A lot of hyperparameters to tune:
  - Optimizers for each component
  - Resetting the components (Discriminator and Classifier)

LGM is cumbersome

- Several terms (8 in total) in the objective:
  - AC-GAN losses: generator loss, generator classification loss, discriminator loss, classifier loss, gradient penalty
  - Generator prototypipcal loss
  - Rehearsal losses: generator rehearsal loss, classifier rehearsal loss
- Complex training scenario
  - Continual learning involves several stages
  - LGM/Classifier training stages
- A lot of hyperparameters to tune:
  - Optimizers for each component
  - Resetting the components (Discriminator and Classifier)
  - AC-GAN vs cGAN

LGM is cumbersome

- Several terms (8 in total) in the objective:
  - AC-GAN losses: generator loss, generator classification loss, discriminator loss, classifier loss, gradient penalty
  - Generator prototypipcal loss
  - Rehearsal losses: generator rehearsal loss, classifier rehearsal loss
- Complex training scenario
  - Continual learning involves several stages
  - LGM/Classifier training stages
- A lot of hyperparameters to tune:
  - Optimizers for each component
  - Resetting the components (Discriminator and Classifier)
  - AC-GAN vs cGAN
  - On which classes should we rehearse (seen or learned)

LGM is cumbersome

- Several terms (8 in total) in the objective:
  - AC-GAN losses: generator loss, generator classification loss, discriminator loss, classifier loss, gradient penalty
  - Generator prototypipcal loss
  - Rehearsal losses: generator rehearsal loss, classifier rehearsal loss
- Complex training scenario
  - Continual learning involves several stages
  - LGM/Classifier training stages
- A lot of hyperparameters to tune:
  - Optimizers for each component
  - Resetting the components (Discriminator and Classifier)
  - AC-GAN vs cGAN
  - On which classes should we rehearse (seen or learned)
  - For how many epochs/steps

LGM is cumbersome

- Several terms (8 in total) in the objective:
  - AC-GAN losses: generator loss, generator classification loss, discriminator loss, classifier loss, gradient penalty
  - Generator prototypipcal loss
  - Rehearsal losses: generator rehearsal loss, classifier rehearsal loss
- Complex training scenario
  - Continual learning involves several stages
  - LGM/Classifier training stages
- A lot of hyperparameters to tune:
  - Optimizers for each component
  - Resetting the components (Discriminator and Classifier)
  - AC-GAN vs cGAN
  - On which classes should we rehearse (seen or learned)
  - For how many epochs/steps
  - Architecture

LGM is cumbersome

- Several terms (8 in total) in the objective:
  - AC-GAN losses: generator loss, generator classification loss, discriminator loss, classifier loss, gradient penalty
  - Generator prototypipcal loss
  - Rehearsal losses: generator rehearsal loss, classifier rehearsal loss
- Complex training scenario
  - Continual learning involves several stages
  - LGM/Classifier training stages
- A lot of hyperparameters to tune:
  - Optimizers for each component
  - Resetting the components (Discriminator and Classifier)
  - AC-GAN vs cGAN
  - On which classes should we rehearse (seen or learned)
  - For how many epochs/steps
  - Architecture
  - etc

LGM via AutoEncoder

Idea #5 Let's extract features for LGM from autoencoder. **Details of training:**

LGM via AutoEncoder

Idea #5 Let's extract features for LGM from autoencoder. **Details of training:**

- Train an AutoEncoder sequentially on each task

LGM via AutoEncoder

Idea #5 Let's extract features for LGM from autoencoder. **Details of training:**

- Train an AutoEncoder sequentially on each task
- For each task, train LGM on the features from AE

LGM via AutoEncoder

Idea #5 Let's extract features for LGM from autoencoder. **Details of training:**

- Train an AutoEncoder sequentially on each task
- For each task, train LGM on the features from AE
- For each task, train Classifier on the features from AE

LGM via AutoEncoder

Idea #5 Let's extract features for LGM from autoencoder. **Details of training:**

- Train an AutoEncoder sequentially on each task
- For each task, train LGM on the features from AE
- For each task, train Classifier on the features from AE
- For each task, rehearse previous data from LGM to LGM

LGM via AutoEncoder

Idea #5 Let's extract features for LGM from autoencoder. **Details of training:**

- Train an AutoEncoder sequentially on each task
- For each task, train LGM on the features from AE
- For each task, train Classifier on the features from AE
- For each task, rehearse previous data from LGM to LGM
- For each task, rehearse previous data from LGM to Classifier

LGM via AutoEncoder

Idea #5 Let's extract features for LGM from autoencoder. **Details of training:**

- Train an AutoEncoder sequentially on each task
- For each task, train LGM on the features from AE
- For each task, train Classifier on the features from AE
- For each task, rehearse previous data from LGM to LGM
- For each task, rehearse previous data from LGM to Classifier
- For each task, rehearse previous data from LGM to AutoEncoder

LGM via AutoEncoder

Idea #5 Let's extract features for LGM from autoencoder. **Details of training:**

- Train an AutoEncoder sequentially on each task
- For each task, train LGM on the features from AE
- For each task, train Classifier on the features from AE
- For each task, rehearse previous data from LGM to LGM
- For each task, rehearse previous data from LGM to Classifier
- For each task, rehearse previous data from LGM to AutoEncoder

Claim: AE features are useless for classification

LGM via AutoEncoder

Idea #5 Let's extract features for LGM from autoencoder. **Details of training:**

- Train an AutoEncoder sequentially on each task
- For each task, train LGM on the features from AE
- For each task, train Classifier on the features from AE
- For each task, rehearse previous data from LGM to LGM
- For each task, rehearse previous data from LGM to Classifier
- For each task, rehearse previous data from LGM to AutoEncoder

Claim: AE features are useless for classification
Justification 1: Joint classification score was only 29% for CIFAR100 in my experiments

LGM via AutoEncoder

Idea #5 Let's extract features for LGM from autoencoder. **Details of training:**

- Train an AutoEncoder sequentially on each task
- For each task, train LGM on the features from AE
- For each task, train Classifier on the features from AE
- For each task, rehearse previous data from LGM to LGM
- For each task, rehearse previous data from LGM to Classifier
- For each task, rehearse previous data from LGM to AutoEncoder

Claim: AE features are useless for classification
Justification 1: Joint classification score was only 29% for CIFAR100 in my experiments
Justification 2: Joint classification score was only 31% for CIFAR100 in Deep InfoMax paper (but their latent code size was smaller (64 dim), which improves classification).

LGM via AutoEncoder (continued)

LGM via AutoEncoder (continued)

Claim: since Classifier cannot learn to classify features, there is no reason to try LGM to learn to generate these class-conditional features.
Justification (hand-wavy): classification is an easier task than generation, so if we can't even classify the objects, there is not hope to learn to generate class-conditional objects.

LGM via AutoEncoder (continued)

Claim: since Classifier cannot learn to classify features, there is no reason to try LGM to learn to generate these class-conditional features.
Justification (hand-wavy): classification is an easier task than generation, so if we can't even classify the objects, there is not hope to learn to generate class-conditional objects.
Idea #6 (didn't try): we need to add classification objective into AutoEncoder's objective

LGM via AutoEncoder (continued)

Claim: since Classifier cannot learn to classify features, there is no reason to try LGM to learn to generate these class-conditional features.
Justification (hand-wavy): classification is an easier task than generation, so if we can't even classify the objects, there is not hope to learn to generate class-conditional objects.
Idea #6 (didn't try): we need to add classification objective into AutoEncoder's objective
Problem: but in this way we won't be able to learn classes one by one since our classification task will be singular (i.e. 1-class classification).

LGM via AutoEncoder (continued)

Claim: since Classifier cannot learn to classify features, there is no reason to try LGM to learn to generate these class-conditional features.
Justification (hand-wavy): classification is an easier task than generation, so if we can't even classify the objects, there is not hope to learn to generate class-conditional objects.
Idea #6 (didn't try): we need to add classification objective into AutoEncoder's objective
Problem: but in this way we won't be able to learn classes one by one since our classification task will be singular (i.e. 1-class classification).
Idea: we can use already learned classes from LGM to fix that

LGM via AutoEncoder (continued)

Claim: since Classifier cannot learn to classify features, there is no reason to try LGM to learn to generate these class-conditional features.
Justification (hand-wavy): classification is an easier task than generation, so if we can't even classify the objects, there is not hope to learn to generate class-conditional objects.
Idea #6 (didn't try): we need to add classification objective into AutoEncoder's objective
Problem: but in this way we won't be able to learn classes one by one since our classification task will be singular (i.e. 1-class classification).
Idea: we can use already learned classes from LGM to fix that
Potential problem: AutoEncoder's performance can deteriorate because of this additional classification objective.

LGM via AutoEncoder (continued)

Claim: since Classifier cannot learn to classify features, there is no reason to try LGM to learn to generate these class-conditional features.
Justification (hand-wavy): classification is an easier task than generation, so if we can't even classify the objects, there is not hope to learn to generate class-conditional objects.
Idea #6 (didn't try): we need to add classification objective into AutoEncoder's objective
Problem: but in this way we won't be able to learn classes one by one since our classification task will be singular (i.e. 1-class classification).
Idea: we can use already learned classes from LGM to fix that
Potential problem: AutoEncoder's performance can deteriorate because of this additional classification objective.
Problem: This will make the whole approach very cumbersome

What was done in February

What was done in February

- Convolutional LGM-VAE on CUB

What was done in February

- Convolutional LGM-VAE on CUB
  - Result: 33% FJA

What was done in February

- Convolutional LGM-VAE on CUB
  - Result: 33% FJA
- Since things didn't work, I started to decompose them and debug each component individually:

What was done in February

- Convolutional LGM-VAE on CUB
  - Result: 33% FJA
- Since things didn't work, I started to decompose them and debug each component individually:
  - Training LGM-VAE, LGM-AC-GAN and LGM-cGAN independently to measure CAS on CUB and MNIST in a joint training scenario

What was done in February

- Convolutional LGM-VAE on CUB
  - Result: 33% FJA
- Since things didn't work, I started to decompose them and debug each component individually:
  - Training LGM-VAE, LGM-AC-GAN and LGM-cGAN independently to measure CAS on CUB and MNIST in a joint training scenario
    - Result: got 25%, 22% and 27% CAS on CUB, 80% vs 93% CAS for LGM-AC-GAN and LGM-cGAN on MNIST (the goal was to get 98%).

What was done in February

- Convolutional LGM-VAE on CUB
  - Result: 33% FJA
- Since things didn't work, I started to decompose them and debug each component individually:
  - Training LGM-VAE, LGM-AC-GAN and LGM-cGAN independently to measure CAS on CUB and MNIST in a joint training scenario
    - Result: got 25%, 22% and 27% CAS on CUB, 80% vs 93% CAS for LGM-AC-GAN and LGM-cGAN on MNIST (the goal was to get 98%).
  - Toy experiments on "Memorizing Networks"

What was done in February

- Convolutional LGM-VAE on CUB
  - Result: 33% FJA
- Since things didn't work, I started to decompose them and debug each component individually:
  - Training LGM-VAE, LGM-AC-GAN and LGM-cGAN independently to measure CAS on CUB and MNIST in a joint training scenario
    - Result: got 25%, 22% and 27% CAS on CUB, 80% vs 93% CAS for LGM-AC-GAN and LGM-cGAN on MNIST (the goal was to get 98%).
  - Toy experiments on "Memorizing Networks"
    - Result: 0 MSE loss for memorizing 10k vectors of size 32 into LSTM.

What was done in February

- Convolutional LGM-VAE on CUB
  - Result: 33% FJA
- Since things didn't work, I started to decompose them and debug each component individually:
  - Training LGM-VAE, LGM-AC-GAN and LGM-cGAN independently to measure CAS on CUB and MNIST in a joint training scenario
    - Result: got 25%, 22% and 27% CAS on CUB, 80% vs 93% CAS for LGM-AC-GAN and LGM-cGAN on MNIST (the goal was to get 98%).
  - Toy experiments on "Memorizing Networks"
    - Result: 0 MSE loss for memorizing 10k vectors of size 32 into LSTM.
  - Training AutoEncoder on CIFAR10/CIFAR100 to incroporate later into LGM

What was done in February

- Convolutional LGM-VAE on CUB
  - Result: 33% FJA
- Since things didn't work, I started to decompose them and debug each component individually:
  - Training LGM-VAE, LGM-AC-GAN and LGM-cGAN independently to measure CAS on CUB and MNIST in a joint training scenario
    - Result: got 25%, 22% and 27% CAS on CUB, 80% vs 93% CAS for LGM-AC-GAN and LGM-cGAN on MNIST (the goal was to get 98%).
  - Toy experiments on "Memorizing Networks"
    - Result: 0 MSE loss for memorizing 10k vectors of size 32 into LSTM.
  - Training AutoEncoder on CIFAR10/CIFAR100 to incroporate later into LGM
    - Result: described on the previous slide.

What was done in February

- Convolutional LGM-VAE on CUB
  - Result: 33% FJA
- Since things didn't work, I started to decompose them and debug each component individually:
  - Training LGM-VAE, LGM-AC-GAN and LGM-cGAN independently to measure CAS on CUB and MNIST in a joint training scenario
    - Result: got 25%, 22% and 27% CAS on CUB, 80% vs 93% CAS for LGM-AC-GAN and LGM-cGAN on MNIST (the goal was to get 98%).
  - Toy experiments on "Memorizing Networks"
    - Result: 0 MSE loss for memorizing 10k vectors of size 32 into LSTM.
  - Training AutoEncoder on CIFAR10/CIFAR100 to incroporate later into LGM
    - Result: described on the previous slide.
  - Training AutoEncoder continually to check its LLL properties
    - Result: it does not forget previous data (it is interesting)

AutoEncoder does not forget previous data

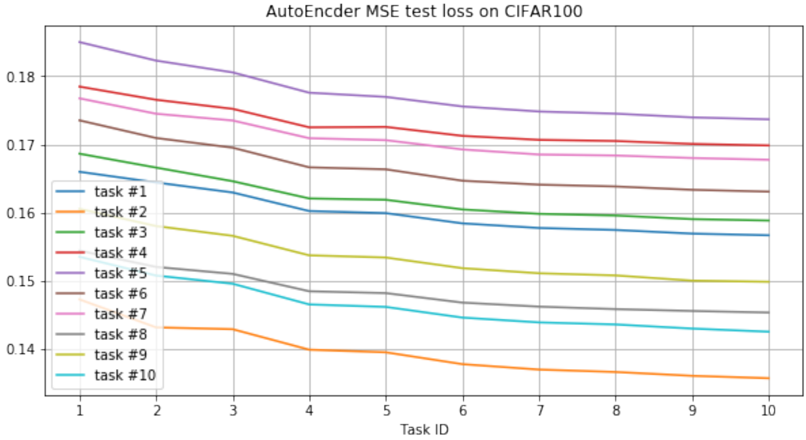


Figure: MSE loss on different tasks for AutoEncoder trained sequentially

About ICCV19 paper

Honestly, there were problems in the paper and how the authors behaved:

About ICCV19 paper

Honestly, there were problems in the paper and how the authors behaved:

- Their approach is cumbersome

About ICCV19 paper

Honestly, there were problems in the paper and how the authors behaved:

- Their approach is cumbersome
- They are not clear about some important details

About ICCV19 paper

Honestly, there were problems in the paper and how the authors behaved:

- Their approach is cumbersome
- They are not clear about some important details
- They try to conceal their research (do not provide the code, do not respond to emails)

That's why I was unconsciously looking for reasons to drop their baseline:

- I think the way the authors do the research is not the way research should be done
- That's why I didn't want to build upon their work (why should we build upon it and spread it if the authors are hiding it?)

Conclusion

To build a good LGM one needs to solve 2 big problems:

- How to build a good feature extractor, i.e. a feature extractor that provides good features for a classifier.
- How to make its features not to drift away

And currently I do not have any concrete ideas to any of these problems.

Some thoughts:

- I do not want to drop the project because it feels like throwing away a 5-month work.
- But I do not see a concrete and principled idea of how to build LGM.
- Current LGM is already too cumbersome and I believe that cumbersome ideas do not survive unless they have outstanding results (like Mask R-CNN)