Continual Learning and Out Of Domain Generalization in Continuous Domain Adaptation

Undergraduate Thesis Report

Submitted in partial fulfillment of the requirements of
BITS F421T Thesis

By

Arshika Lalan
ID No. 2017B3A70620G

Under the supervision of:

Prof. Gabriel Kreiman
&
Prof. Hanspeter Pfister

KREIMAN LAB
November 2023

Chapter 1

Continual Learning

1.1 Continual Learning and Catastrophic Forgetting

To achieve general artificial intelligence, it is imperative that agents have the ability to learn and remember a myriad of different tasks and perform reasonably well on them. This is a nuanced problem in the real-word: quite often, tasks will not be sequentially labelled and the amount of training examples for one particular task may be less. Tasks may also switch between themselves unpredictably. Intelligent agents must demonstrate a capacity for continual learning. The agents must be able to learn new tasks without forgetting the information gained to perform previous tasks.

Catastrophic forgetting is a phenomenon that is a significant challenge in achieving continual learning. It occurs when the network is trained sequentially on multiple tasks because the weights in the network that are important for task A are changed to meet the objectives of task B.

Alleviating catastrophic forgetting is crucial in implementing algorithms for achieving catastrophic forgetting.

1.2 Experiment 1

Task: Evaluation of catastrophic forgetting problem by taking a network and assessing performance upon sequential training on tasks.

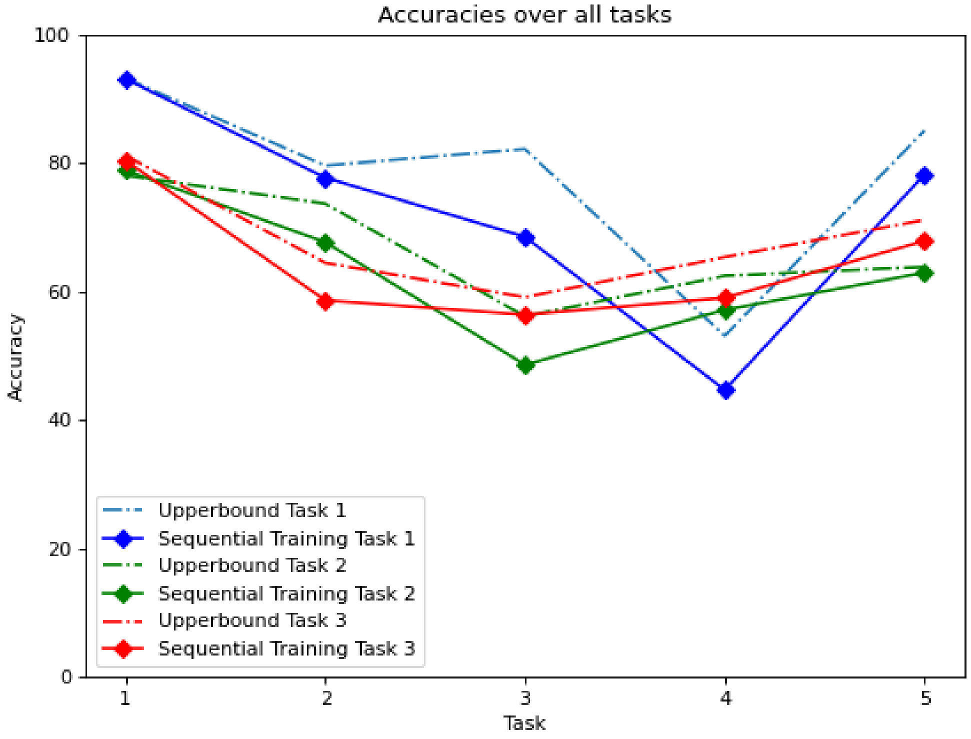


FIGURE 1.1

1.2.1 Setup

I used the CIFAR 10 dataset and the ResNet architecture for assessing the impact of sequential training. There were 5 tasks from Task 1 (categories 1, 2) upto Task 5 (categories 9, 10) I conducted the following three experiments:

- Sequentially trained ResNet from Task 1 to Task 5 while testing on Task 1 in between training sessions.
- Shuffled the Task on which testing was done (i.e testing on Task 2 as opposed to Task 1)
- Calculating the Upper Bound for the test Task.

I tested on 3 Tasks - Task 1 (category 'airplane' vs category 'car'), Task 2 (category 'bird' vs category 'cat') and Task 3 (category 'deer' vs category 'dog').

1.2.2 Results

I obtained the graph 1.1 on sequential training. The process went as follows. I trained the first task (Task 1) and then tested the ResNet on Task 1. I proceeded to then train ResNet on Task 2 and Test on Task 1, followed by training on Task 3 and testing on Task 1 and so on. When I repeated the experiment with testing on Task 2, the first sequential task above is Task 2. Thus, the process for the green line is training on Task 2, followed by testing on Task 2, training on Task 3 followed by testing on Task 2 uptil Training on Task 1 and Testing on Task 2.

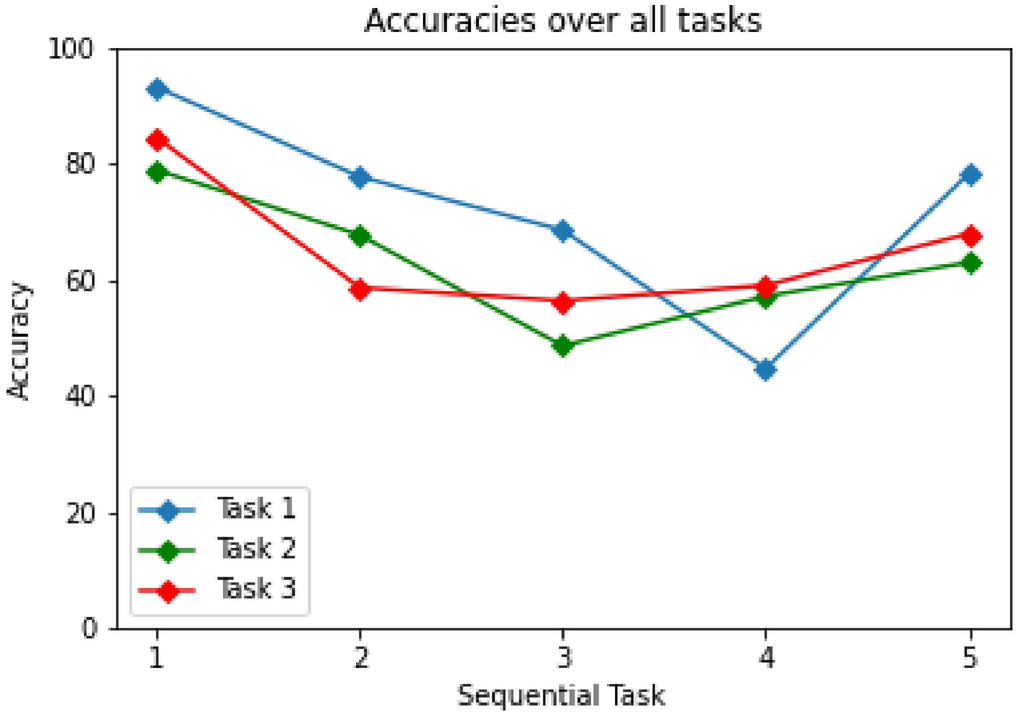


FIGURE 1.2

1.3 Review of existing algorithms

1.3.1 EWC [2]

- This algorithm slows down learning on certain weights based on how important they are to previously seen tasks.

- There are many possible configurations of weights and biases for task B that will give low performance. This is key to EWC. While learning task B, EWC therefore protects the performance in task A by constraining the parameters to stay in a region of low error for task A.

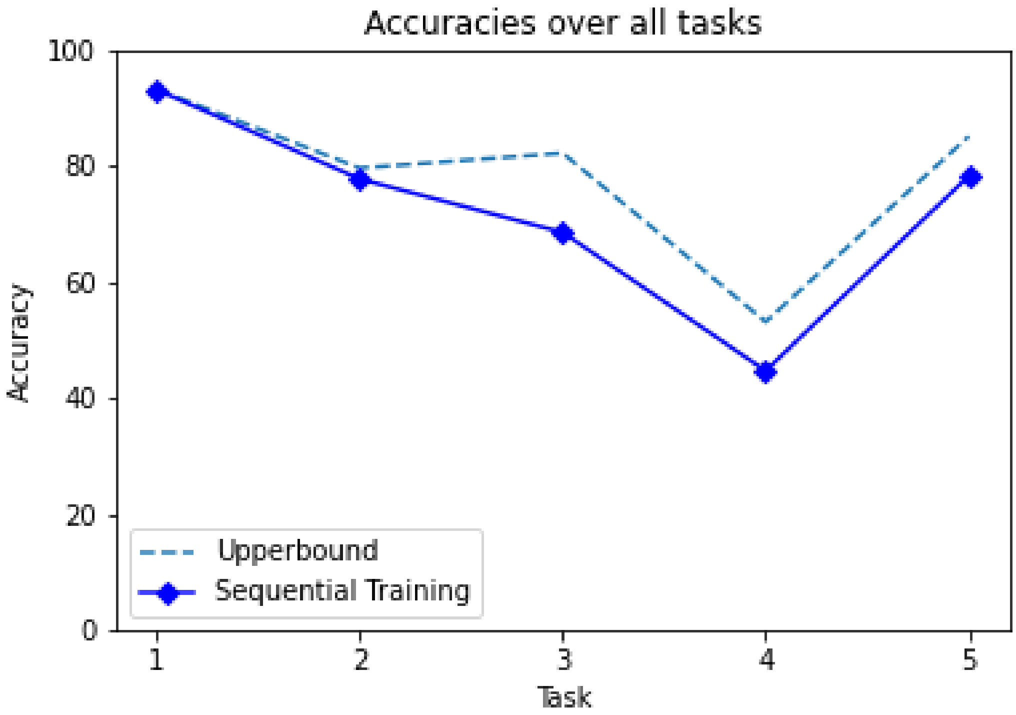


FIGURE 1.3

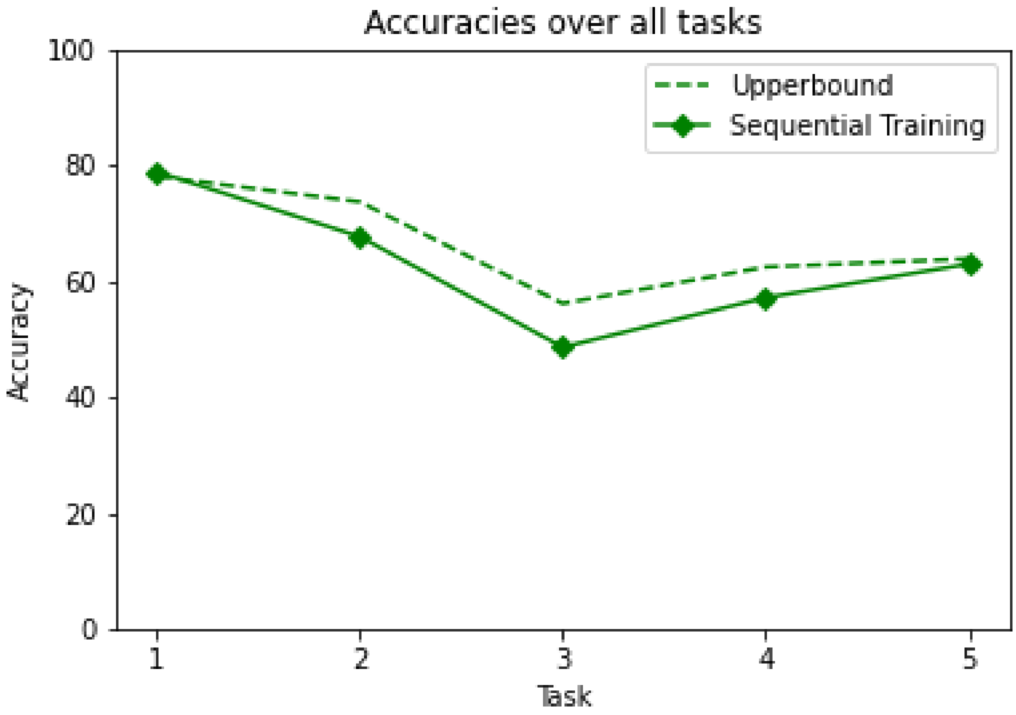


FIGURE 1.4

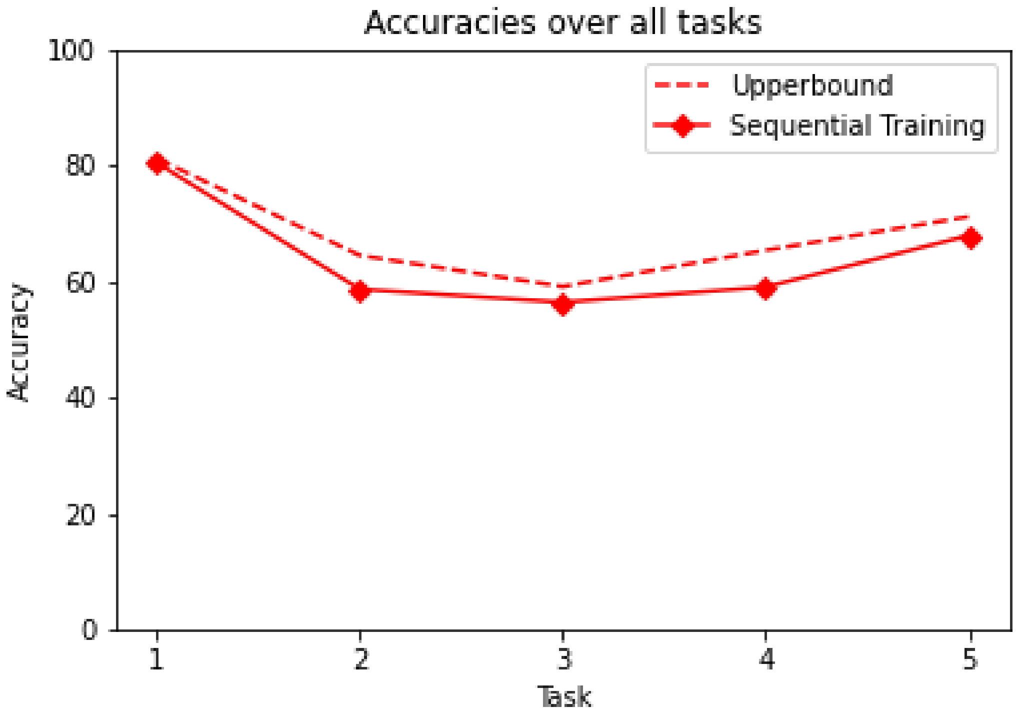


FIGURE 1.5

- DRAWBACK: Exactly computing the diagonal of the Fisher requires summing over all possible output labels and thus has complexity linear in the number of outputs. This limits the application of this approach to low-dimensional output spaces.

SUMMARY: EWC algorithm prevents catastrophic forgetting by taking inspiration from biological synaptic connections to reduce their plasticity on learning new tasks. EWC slows down learning for the important weights of the previously learned tasks. This gives better performance than L2 regularization, which constrains all weights equally and significantly reduces learning on new tasks. EWC treats the parameters as probabilistic distributions and assumes all relevant information for the previously learned task is in the posterior probability of the parameters given the previous data.

1.3.2 SI [5]

- Greatest disparity between ANNs and the biological neural network lies in the complexity of the synapses.
- In ANNs, individual synapses (weights) are typically described by a single scalar quantity. On the other hand, individual biological synapses make use of complex molecular machinery that can affect plasticity at different spatial and temporal scales

- Simple scalar 1D synapses suffer from catastrophic forgetting
- Usage of 3D state space to alleviate catastrophic forgetting
- In our model, the synaptic state tracks the past and current parameter value, and maintains an online estimate of the synapse’s “importance” towards solving problems encountered in the past

- In our model, the synaptic state tracks the past and current parameter value, and maintains an online estimate of the synapse’s “importance” towards solving problems encountered in the past

SUMMARY: SI algorithm is similar to EWC in that it penalizes the update of the weights (”synapses”) that are the most important to previous tasks. However, while EWC does this in an online fashion after the completion of the training task, SI does this in an online fashion over the entire learning trajectory. The results from EWC and SI are comparable over the split MNIST dataset.

1.3.3 GEM [3]

- GEM leverages an episodic memory to avoid catastrophic forgetting and favors positive backward transfer.

- GEM leverages an episodic memory to avoid catastrophic forgetting and favors positive backward transfer.

- Memory: m = M/T samples of all tasks are kept in the memory. These samples are the most recent ones in the paper.
- Episodic: Replay over the memories to make sure accuracy on previous tasks is not damaged as we learn new tasks
- Gradient: Checking that the gradient doesn’t run the wrong way on learning tasks, i.e. we do not unlearn previously learned tasks.
- Gradient update rule: Constrain your update for current task to not conflict with update for the previous task. If the gradient is going the wrong way, then take the vector and project it to the closest gradient that doesn’t go the wrong way. The optimization is formulated as a projection onto a cone.
- DRAWBACKS: GEM assumes access to task boundaries and an i.i.d. distribution within each task episode. It divides the memory budget evenly among the tasks. i.e. m = M/T slots is allocated for each task, where T is the number of tasks. The last m examples from

each task are kept in the memory. This has clear limitations when the task boundaries are not available or when the i.i.d. assumption is not satisfied.

- Practical use of GEM is impeded by several limitations: (1) the data examples stored in the episodic memory may not be representative of past tasks; (2) the inequality constraints appear to be rather restrictive for competing or conflicting tasks; (3) the inequality constraints can only avoid catastrophic forgetting but can not assure positive backward transfer.

SUMMARY: GEM leverages an episodic memory to avoid catastrophic forgetting. It provides negligible to positive forward transfer and favors positive backward transfer, something which does not happen with algorithms like EWC and SI. However, GEM does not leverage task descriptors to aid zero-shot learning.

1.3.4 GSS [1]

- Formulates sample selection as a constraint reduction problem based on the constrained optimization view of continual learning.
- Prior focused: The parameter gradually drifts away from the feasible regions of previous tasks, especially when there is a long chain of tasks and when the tasks resemble each other
- The replay-based approach stores the information in the example space either directly in a replay buffer or in a generative model

SUMMARY: GSS uses replay to enhance continual learning. Parameter gradients are minimized over the memory buffer M of samples. Selection of memory buffer becomes an important research problem. A greedy algorithm is also proposed to enhance efficiency

Chapter 2

Continuous Domain Adaptation

2.1 Continuous Domains

In real life, many domains have continuous features or actions. For example, the various phases of day and night, or the continuous range of human age. Continual learning is especially important and practical in continuous domains, since most domains in the practical world are continuous. Imagine a tumor detection system for lungs trained on lungs of patients in their mid-twenties. Its efficacy may be severely impacted if tested on lungs of patients who are sixty plus. Such a scenario is called a domain shift. A visual recognition must adapt to different domain shifts and preserve accuracy.

2.2 Asymmetry in efficacy on domain shifts

To understand whether there is any asymmetry in accuracy when the start points of the two domains are flipped, the following experiment was carried out.

The continuous domain is chosen to be the rotation of an object along the z-axis. To simulate a continuous domain, a step size of 5 degrees rotation was chosen to generate the images. After generation, every step size was divided into a continual learning task.

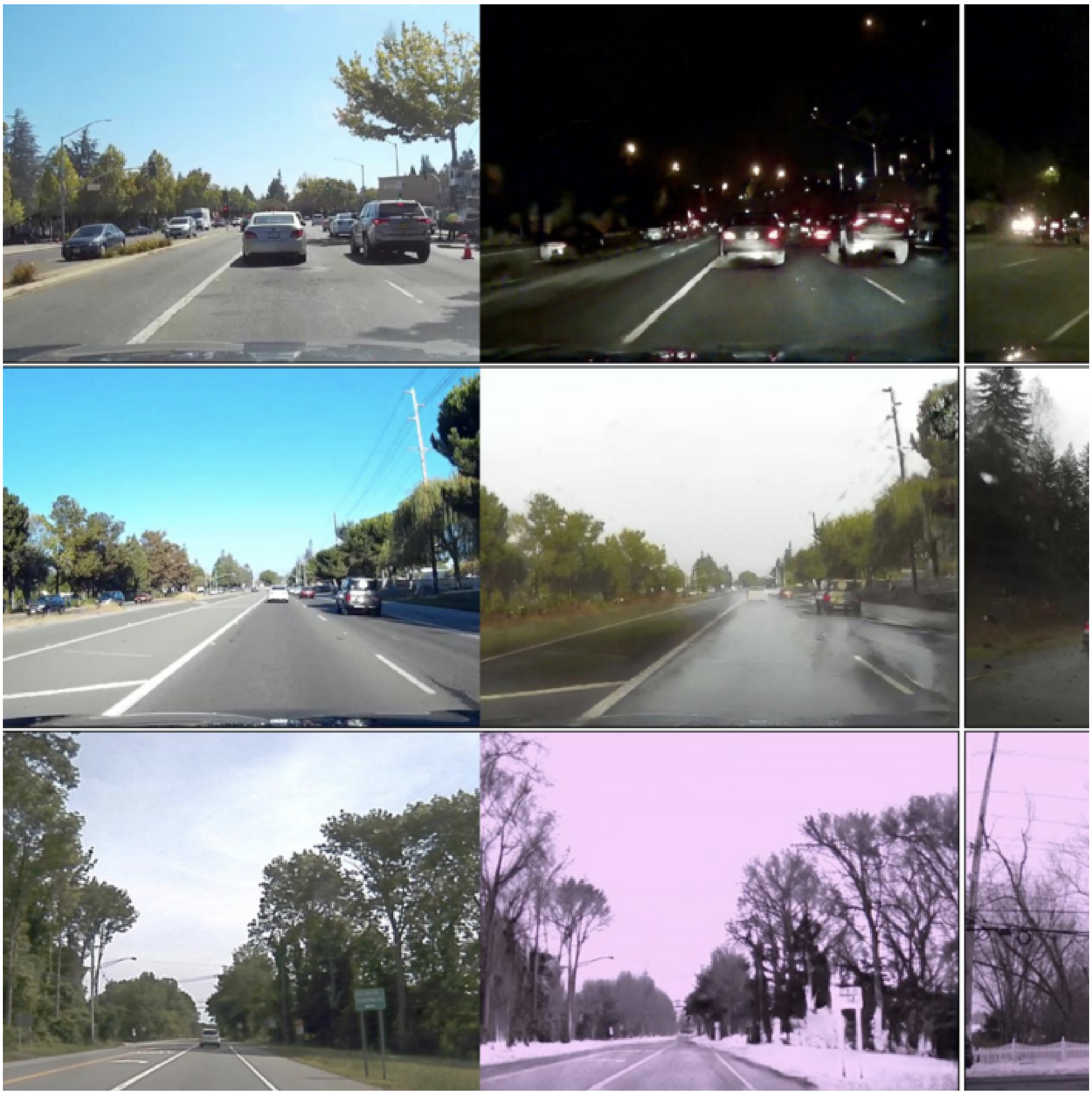


FIGURE 2.1: Source: Wang, H., He, H., Katabi, D. (2020). Continuously indexed domain adaptation. arXiv preprint arXiv:2007.01807. [4]

2.3 Blender

To generate the images for training, Blender was used. Blender is a free and open source 3D rendering software, and is popularly used to generate life-like objects that can be used to create machine learning datasets.

2.4 ShapeNet

ShapeNet is a richly annotated dataset of 3D shapes. It enables research in computer graphics, computer vision, robotics, and other related disciplines. ShapeNet is a collaborative effort between researchers at Princeton, Stanford and TTIC. The ShapeNet objects were imported in

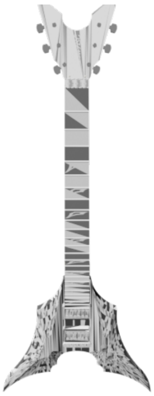
Blender, and 36 rotated views from 0 degrees to 180 degrees were produced. At every view, 20 random scaling radii were applied drawn from a uniform distribution to increase the complexity of the dataset.

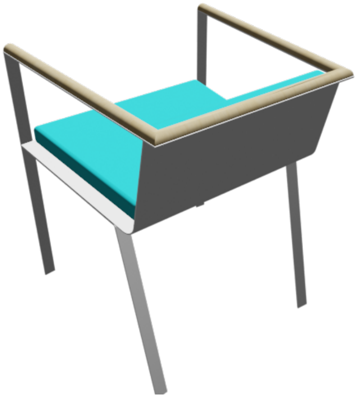
2.5 PseudoCode

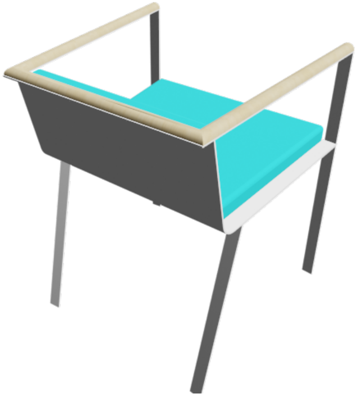
```
# Preprocessing
-> Generate 36 views (180/5) for each instance of each class with camera-distance variation
/scaling variation using np.random()
-> Split the generated instances into test-train (30:10)

# Training
-> For every 5 degree stepsize define 1 task , total 36 tasks
-> For task i in num_tasks:
     -> Shuffle all available instance images and train them on a CNN
     -> Test CNN on task i, record accuracy
     -> Test CNN on tasks [0,i-1], record accuracy
-> Restart training but flip the order of tasks and record accuracies
-> Average out accuracies over all class instances and plot
```

2.6 Renders







Bibliography

[1] Rahaf Aljundi et al. “Gradient based sample selection for online continual learning”. In: *arXiv preprint arXiv:1903.08671* (2019).

[2] James Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[3] David Lopez-Paz and Marc’Aurelio Ranzato. “Gradient episodic memory for continual learning”. In: *Advances in neural information processing systems* 30 (2017), pp. 6467–6476.

[4] Hao Wang, Hao He, and Dina Katabi. “Continuously indexed domain adaptation”. In: *arXiv preprint arXiv:2007.01807* (2020).

[5] Friedemann Zenke, Ben Poole, and Surya Ganguli. “Continual learning through synaptic intelligence”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3987–3995.