**Variational Continual Learning in Deep Discriminative Models**
Gwangbin Bae (gb585), Riccardo Barbano (rb876), Justin Bunker (jb2200)

### Continual Learning
- Data may arrive in non i.i.d. way
- Tasks may change and/or new tasks may emerge

**Plasticity vs. Stability**
- Catastrophic forgetting (inability to adapt)

### Bayesian Inference
BI provides a framework for continual learning
\[
p(\theta|D_{1:T}) \propto p(\theta) \prod_{i=1}^{T} p(D_i|\theta)
\]
\[
\propto p(\theta|D_{1:T-1}) p(D_T|\theta)
\]

**Posterior Update:** normalize (current posterior $\times$ likelihood of newly observed data)

### Projection Operation
PO finds a tractable normalized distribution that approximates the intractable un-normalized posterior
\[
p(\theta|D_{1:T}) \approx q_T(\theta) = \text{proj}(q_{T-1}(\theta)p(D_T|\theta))
\]
- Recursive relation between posteriors recovered
- $q_0(\theta) = p(\theta)$ (initialized with prior distribution)

### Variational Continual Learning (VCL)
VCL uses KL divergence minimization for projection
\[
q_\ell(\theta) = \arg \min_{q \in Q} KL \left( q(\theta) || \frac{1}{Z_\ell} q_{\ell-1}(\theta)p(D_\ell|\theta) \right)
\]
- $Q$: set of available posterior functions
- $q_\ell(\theta)$: Gaussian mean-field approximate posterior

### Episodic Memory Enhancement
**Coreset**
- Small subset of data from each task
- Excluded in training & used before prediction
- Avoids catastrophic forgetting

**Coreset Heuristics**
- Random selection & K-center methods

### Multi-head Network
- Standard architecture for multi-task learning

**Head Network**
- Task-specific update
- Baseline: 1 layer

**Shared Network**
- Continuous update
- Baseline: 2 layers

**Posterior Update:** finds $q_\ell(\theta)$ that maximizes the negative variational online free energy
\[
\sum_{N=1}^{N_T} \mathbb{E}_{\theta \sim q_\ell(\theta)} \left[ \log p \left( \tilde{y}_i^{(n)}|\tilde{x}_i^{(n)}; \theta \right) \right] - KL(q_\ell || q_{\ell-1})
\]

**Expected Likelihood** adapts to the new task
- Intractable & approximated using Monte Carlo sampling and local reparameterization trick

**KL Divergence** avoids forgetting previous tasks
- Tractable & $q_\ell(\theta)$ initialized with small variance

### Algorithm

```
1 for t = 1, ..., T # T: total no. tasks
2 Observe new task $D_t$
3 if first task
4 # train feed-forward NN with $D_t$
5 Initialize $q_0(\theta)$ with MLE
6 Update coreset $C_t$ with $C_{t-1}$ and $D_t$
7 # update posterior
8 $q_{\ell}(\theta) = \text{proj}(q_{\ell-1}(\theta), D_t U C_{t-1} C_t)$
9 for task in previous tasks
10 # incorporate coreset
11 $q_{\ell}(\theta) = \text{proj}(q_{\ell}(\theta), C_t)$
12 # make prediction
13 get_score(test_x, test_y, $q_{\ell}(\theta)$)
```

### Experiments & Results

#### 1. Plasticity vs. Stability (dataset: Split MNIST)
- **Likelihood only** → catastrophic forgetting
- **KL only** → inability to adapt

#### 2. Performance vs. Coreset Size

#### 3. Task Accuracy (Split MNIST & Split NotMNIST)

#### 4. Extensions
- Changing the incoming task order
- Changing the number of shared/head layers

### Conclusion
Main contribution of this project includes:
- Customizable implementation of VCL pipeline in PyTorch
- Network architecture, task ordering, etc.
- Demonstration of various consequences from changing model characteristics
- Performance increase with episodic memory

### References