Accelerating Deep Learning Inference via Freezing

Adarsh Kumar, Arjun Balasubramanian, Shivaram Venkataraman, Aditya Akella
Deep Learning – State of affairs

Over the years
- Top 5 error rate decreasing
- Models becoming deeper

Top Competitors - ImageNet Large Scale Visual Recognition Challenge
Deep Learning – State of affairs

Over the years
- Top 5 error rate decreasing
- Models becoming deeper

Suits goals for ML training

Top Competitors - ImageNet Large Scale Visual Recognition Challenge
Deep Learning – State of affairs

Over the years
- Top 5 error rate decreasing
- Models becoming deeper

Suits goals for ML training

Not aligned with goals for ML inference

Top Competitors - ImageNet Large Scale Visual Recognition Challenge
Deep Learning - Background

Neural Network - Sequence of layers with each layer dependent on previous layers
Deep Learning – State of affairs

Not aligned with goals for ML inference
- Requires low latency
- Challenge due to deeper models

Top Competitors - ImageNet Large Scale Visual Recognition Challenge

Image source: Google Images
Deep Learning – Reducing Latency

Prior Solutions

• **Model Quantization:** Changes precision of computation; **Hurts accuracy**
• **Model Distillation:** Smaller model is trained to mimic larger/ensemble model; **Hurts accuracy**
• **Ensemble Methods:** Run multiple models, choose best; **Resources wasted**
• **Anytime Predictions:** Auxiliary Predictions; **Trade-off b/w accuracy and latency**
• **Custom Hardware:** TPUs, FPGAs; **Hardware dependent**
Freeze Inference

Provides low-latency inference by caching intermediate layer outputs

Goals

• No trade-off on accuracy
• Resource efficient
• Hardware agnostic
Freeze Inference – Key Insight

- **Input Layer**
- **Background Subtraction**
- **Edge Detection**
- **Output Layer**

**JUMBO**

![Diagram](image)

- Input to layer is not same for both images
- Input to layer is same for both images
Freeze Inference – Basic Mechanism

Prior to Inference - Cache intermediate layer outputs
Freeze Inference – Basic Mechanism

During Inference – Look-up from cache

Input $X_1$ \rightarrow L_1 \rightarrow L_2 \rightarrow L_K \rightarrow L_N \rightarrow \text{Output Layer} \rightarrow \text{Prediction } Y

INTERMEDIATE OUTPUT CACHE

Input $X_2$ \rightarrow L_1

CACHE MISS
CACHE HIT
Freeze Inference – Basic Mechanism

During Inference – Look-up from cache

INTERMEDIATE OUTPUT CACHE

Input $X_1$ -> L_1 -> L_2 -> \ldots -> L_K -> L_N -> Output Layer -> Prediction Y

Input $X_2$ -> L_1 \rightarrow L_2

CACHE MISS

CACHE HIT
Freeze Inference – Basic Mechanism

During Inference – Look-up from cache
Freeze Inference – Basic Mechanism

During Inference – Look-up from cache

![Diagram showing the process of freeze inference with cache hit and miss]

- **Input** $X_1$ is fed into the network.
- Intermediate outputs are stored in the cache.
- During inference, the network looks up from the cache for previously computed intermediate outputs.
- If the required output is in the cache, it is a **cache hit**.
- If not, it is a **cache miss**.
- The final output $Y$ is predicted based on the complete network computations.
Freeze Inference – Basic Mechanism

During Inference – Look-up from cache

![Diagram showing the basic mechanism of Freeze Inference during inference, including look-up from cache.](image)
Freeze Inference - Challenges

Challenge #1: Exact cache hit is unlikely

Challenge #2: Curse of dimensionality

Challenge #3: Memory and computational overheads
Freeze Inference - Challenges

Challenge #1: Exact cache hit is unlikely

Challenge #2: Curse of dimensionality

Challenge #3: Memory and computational overheads
Freeze Inference - Challenges

Challenge #1: Exact cache hit is unlikely

Why?
- High dimensions
- Floating point precision
Freeze Inference - Challenges

Challenge #1: Exact cache hit is unlikely

Why?
- High Dimensions
- Floating point precision

Approach?
- Points close by in feature space have high probability of having same prediction
Freeze Inference - Challenges

Challenge #1: Exact cache hit is unlikely

Towards Approximate Caching
Instead of exact matches, find “k” nearest points in the cache

Prediction is the majority label among the “k” nearest neighbors
Freeze Inference - Challenges

Challenge #1: Exact cache hit is unlikely

Need ”confidence” to infer the quality of a prediction

Prediction can be more “confident” if:
(i) More neighbors agree on the same label
(ii) Neighbors are closer to the input point
Freeze Inference - Challenges

Challenge #1: Exact cache hit is unlikely

Need ”confidence” to infer the quality of a prediction

Prediction can be more “confident” if:

(i) More neighbors agree on the same label

(ii) Neighbors are closer to the input point
Freeze Inference - Challenges

Challenge #1: Exact cache hit is unlikely

Need ”confidence” to infer the quality of a prediction

Prediction can be more “confident” if:
(i) More neighbors agree on the same label
(ii) Neighbors are closer to the input point

Confidence - Heuristic based on the above
Freeze Inference - Challenges

Challenge #1: Exact cache hit is unlikely

Towards Approximate Caching
How much confidence is good enough?
- Need to establish a “threshold” per layer.

Validation dataset → Each point
Make prediction at layer “k” → Correct prediction?

No
Threshold at layer “k” = Max. such observed confidence
Freeze Inference - System Design
Freeze Inference - Results

Evaluation against -
- Datasets: CIFAR-10 and CIFAR-100
- Models: ResNet-18 and ResNet-50

For each test,
- Use 35,000 points for cache construction
- Use 5,000 points for threshold computation
- Apply Freeze Inference for 10,000 requests
Freeze Inference – Results

Upper Bound

ResNet-18 results

Actual
Freeze Inference – Results

Block 5 – k-NN: ~25%  Upper bound: ~90%
Bridging this gap is an interesting research problem
Freeze Inference – Discussion

Discussion Point #1 – Managing memory requirement
- Storing each point incurs memory overheads
- Can use k-means to reduce memory overheads
- Given a fixed cache budget $M$, choose points to constitute cache
Freeze Inference – Discussion

Discussion Point #2 – Cache placement
- To be placed closed to region of compute for low latency
- Cache placement on GPUs
Conclusion

Can use caching of intermediate layer outputs to reduce inference latency

Open research challenges to fully realize the potential

- Adaptation to custom hardware like GPUs
- Computational and memory overheads
- Online cache construction mechanism
- Better cache look-up schemes
Backup Slides
Freeze Inference - Challenges

Challenge #2: Curse of Dimensionality

- Distance based similarity measures do not work well in high dimension
- **Impact:** Cache look-up will not be accurate
Freeze Inference - Challenges

Challenge #2: Curse of Dimensionality

- Distance based similarity measures do not work well in high dimension
- **Impact:** Cache look-up will not be accurate

**Solution?**

- Inspired by metric learning, use a one layer neural network for supervised dimensionality reduction
Freeze Inference - Challenges

Challenge #3: Memory and Computational Overheads

k-nearest neighbors necessitates -
- Compute: Distance to be computed against each point in cache
- Memory: To hold the cache
Freeze Inference - Challenges

**Challenge #3: Memory and Computational Overheads**

k-nearest neighbors necessitates -
- Compute: Distance to be computed against each point in cache
- Memory: To hold the cache

**Solution?**

Can use k-means to cluster points in cache
Store only cluster centers and associated labels in cache
Freeze Inference - Results

Memory overheads depend on –
(i) # layers in model
(ii) Lower dimension size (d)
(iii) Value of “k” in k-NN

<table>
<thead>
<tr>
<th>Model</th>
<th>Memory (d=1024 and k=100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>12.5 MB</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>25 MB</td>
</tr>
</tbody>
</table>
Freeze Inference – Discussion

Discussion Point #3 – Online Cache Updates
- Incorporating inference points into cache
- Handling frequent inference queries