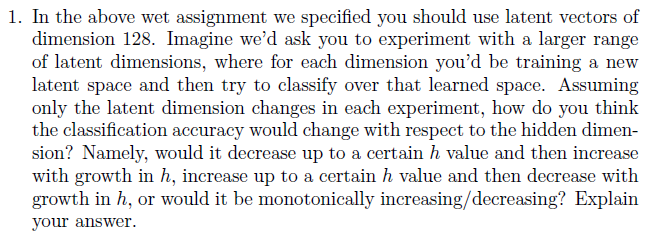
{ Final Project – Dry }

Submitted by :

|  |  |  |
| --- | --- | --- |
| Name | ID | Email |
| Husain Azaqy | 213382104 | Hssain.azaqy@campus.technion.ac.il |
| Tomer Danilin | 207933961 | tomerdanilin@campus.technion.ac.il |

Note: For checking convenience , the questions where copied to this file (Black frames)



**Answer:**

We expect the classification accuracy to peak at a certain h value, from there the accuracy would either plateau (no improvements will be made) or even drop for way too large h since classifying over a larger latent space should be harder beyond some point ,due to the classifier’s compute being fixed in this context.

Since the latent space is a learned alternative representation of the signal domain, given a fixed encoder architecture, we expect that up to some h the learned latent space will get richer, beyond that h we think the learned representation can have redundancies and even get too complex to classify over.

A close up of a text

AI-generated content may be incorrect.

**Answer:**

1. UAT: for any loss criterion that is reasonable **(\*\*\*)** , there exists a finite width fully connected neural network of depth 1 with activation that is non -polynomial, bounded and continuous such that:

Is Optimal.

**(\*\*\*):** Theloss is continuous ,Proper (Consistent) meaning as the network output gets closer to the true function, the loss should approach zero. The loss is Differentiable (or Subdifferentiable) and is Convex.

These properties ensure that training a neural network is stable, efficient, and leads to meaningful learning. Yet some assumptions can be ignored in practice.

This theorem directly supports the claim of the friend in question. Note that this is the general format of the theorem, the original theorem is for uniform convergence (supremum of differences between outputs approaches zero).

1. The conclusion is wrong due to many practical reasons:
2. The theorem provides an exponential bound on the width of the single layer network, which is largely impractical. That is due to optimization and memory limits. This is known as the curse of dimensionality.
3. Not only it is possible to “compensate” for lack of network width by depth, the added activations between layers help find more complex relations more efficiently. And the structure of the relations learned by each layer are more hierarchical and explainable. Early layers tend to learn simple features while later layers tend to extract more complex and robust features and relations.

A white text on a white background

AI-generated content may be incorrect.

**Answer:**

1. The advantages of using a CNN over an MLP are as follows:

|  |  |  |
| --- | --- | --- |
| Criterion | CNN | MLP |
| Parameter count | Uses shared weights and local connections, reducing parameters. | Fully connected layers make them memory heavy. |
| Translation invariance or equivariance | Recognize objects regardless of position due to pooling and convolution. | Lack this property and are sensitive to position changes. |
| Spatial considerations | Automatically extract spatial features like edges and patterns. Have a receptive field that scales with depth. | Treat all pixels as independent, ignoring spatial relationships. |
| Computational Efficiency | Faster due to weight sharing in the form of filters/kernels and generally lower hidden dimensions. | Requires more memory and computation. |
| Scalability to Large Images | Efficiently process high-dimensional images. | Become impractical as image size increases. |

1. We don’t agree with Alice , like mentioned above, yes the convolution operation is a linear operation, but it is applied differently, for each filter, the weights are universal no matter where it is applied on the image. This is because it is applied with a sliding window fashion which allows it to capture more spatial features.

Compared to an MLP, which seemingly treats all pixels as independent ignoring any spatial relationships.

A close up of text

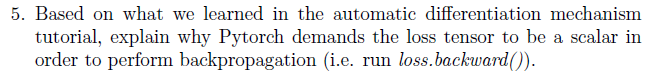
AI-generated content may be incorrect.

**Answer:** We compare both approaches as follows:

|  |  |  |
| --- | --- | --- |
| Aspects | Exponential Moving Average | Linear Averaging |
| Responsiveness to recent updates in gradients | More responsive to recent gradients due to exponential decay of past gradients. | Less responsive, as the decay is linear and gradual. |
| Memory limitations | Requires storing only the current and previous values. | Requires storing all past gradients within the window. |
| Convergence speed and efficiency | Faster convergence due to better gradient adaptation. | Slower convergence, not as efficient due to slower adaptation. |
| Adaptation of learning rate | Allows for dynamic adjustment of learning rates (e.g., in Adam). | Less effective at adapting learning rates. |

To sum things up:

An exponential moving average is generally better for efficiency, convergence, and more responsive, while linear moving average tends to be less adaptive to recent gradients and slower for convergence.



**Answer:**

The loss tensor should be a scaler due to the recursive nature of the reverse mode of AD.Since in the reverse mode of AD we compute the gradients relevant to each node in the computational graph in reverse topological order. We multiply the previous derivative with the current one and update the current gradient. We need the multiplication to always compile therefore the loss is a scaler. If the loss was a matrix or a vector the derivative would be a matrix making the multiplications in this algorithm not possible.

A close up of text

AI-generated content may be incorrect.

**Answer:**

A white text with black text

AI-generated content may be incorrect.

**Answer:**

A close up of a map

AI-generated content may be incorrect.

A screenshot of a computer game

AI-generated content may be incorrect.

**Answer:**

A white text with black text

AI-generated content may be incorrect.

**Answer:**

A text on a page

AI-generated content may be incorrect.

**Answer:**