# PROTOTYPE MEMORY AND ATTENTION MECHANISMS FOR FEW SHOT IMAGE GENERATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Recent discoveries indicate that the neural codes in the primary visual cortex (V1) of macaque monkeys are complex, diverse and sparse. This leads us to ponder the computational advantages and functional role of these “grandmother cells". Here, we propose that such cells can serve as prototype memory priors that bias and shape the distributed feature processing during the image generation process in the brain. These memory prototypes are learned by momentum online clustering and are utilized via a memory-based attention operation. Integrating this mechanism, we propose *Memory Concept Attention* (**MoCA**) that aims to improve few shot image generation quality. We show that having a prototype memory during attention can improve image synthesis quality, learn interpretable visual concept clusters, as well as improve the robustness of the model. Insofar as GANs can be seen as plausible models for reasoning about the top-down synthesis in the analysis-by-synthesis loop of the hierarchical visual cortex, our findings demonstrate a plausible computational role for these “prototype concept" neurons in visual processing in the brain.

## 1 INTRODUCTION

Recent neurophysiological findings based on calcium imaging have revealed that many neurons in V1 are strongly tuned to complex local patterns, rather than simple oriented edges or Gabor wavelets. These complex neurons exhibit far stronger responses (a 3-5 fold increase) when exposed to their preferred patterns than when exposed to simple grating and bar-like stimuli. The high degree of specificity in these neurons’ selectivity suggests that they might serve as specific pattern detectors or a type of "grandmother neuron" [28]. Due to the selectivity towards complex stimuli, the population response of these V1 neurons are extremely sparse w.r.t. any given stimulus. Only 4–6 neurons out of roughly 1000 neurons would respond strongly to any given pattern or natural stimulus [29]. This finding is reminiscent of an earlier study that found similarly sparse encoding of concepts in the hippocampus [23]. It is possible such "grandmother cells" are prevalent at every level of the visual hierarchy. These observations raise the following question: what are the possible computational benefits and rationales of having such a diverse set of sparse-responding complex feature detectors in the early visual cortex?

In this paper, we investigate the usefulness of visual concept prototype neurons in the context of image generation. Image synthesis is a central theme in a number of hierarchical models of the visual systems, including interactive activation and predictive coding [9, 17, 21, 22, 24], and is hypothesized to take place in the top-down feedback connections between areas in the visual cortex. We propose that these "grandmother neurons" serve as representative prototype semantic cells that can retrieve their corresponding pools of component cells in the memory. The lateral connections between prototype semantic cells introduces competition that results in sparse activation. We push this sparsity to its extreme with a winner-take-all strategy in which only a single prototype semantic cell is activated in each hypercolumn. Upon excitation, the component cells associated with a given prototype semantic cell are retrieved from memory and used as a structural prior, which can augment a novel activation with previously learned patterns. These priors allow the synthesis process to look beyond the current spatial context and utilize the prototype memories learned and accumulated over time. In our model, these prototype semantic cells and their corresponding component cells in the memory are dynamically updated using a momentum encoder. Each prototype semantic cell is computed using a permutation invariant function of its corresponding component cells, while component cells

are updated during training with fresh contextual information. This online hierarchical update loop facilities the development of cluster structures while maintaining high computational efficiency. Thus, the "grandmother cells" can be considered as a key to form the structural conceptual priors in a memory attentive process during image generation.

To emphasize the importance of the attention operation beyond single image activation, we name our proposed memory-based attentional process *Memory Concept Attention* (**MoCA**). It can be defined as a single layer that can be plugged into existing generator architectures in the GAN framework. We test our model by performing extensive experiments using state-of-the-art StyleGAN2 [13] and newly proposed few-shot image generator FastGAN [19]. Our experiments demonstrate the utilization of information within the semantic clusters accumulated during training can improve few shot image generation task on Animal-Face Dog [27], 100-Shot-Obama [40], ImageNet-100 [25], COCO-300 [18], CIFAR10 [16] and Caltech-UCSD Birds (CUB) [35]. Additionally, we also find that generators with MoCA can resist a certain degree of noise injection during testing time, suggesting that using attention in a structured memory prior can allow a more robust image synthesis process. Overall, this study provides insights into the potential computational role of the "grandmother" neurons in the cortex.

## 2 Related Work

**Visual concept learning** Visual concepts, when used in a feedforward voting scheme, have been shown to be effective in overcoming misclassification due to occlusion [33]. Explicit representation of visual concepts using prototype neurons might also serve as effective reconfigurable parts for building compositional machines [1, 6, 41]. In this paper, we explored the use of visual concepts in the form of prototype memories in an attentional mechanism, which might extend its application in more complex compositional tasks such as image generation.

**Self-attention** The attention mechanism [31] in deep learning is widely popular in the Natural Language Processing (NLP) community and is also known as non-local networks in the vision community [34]. [38] introduced self-attention into generative models in the Self-Attention GAN. Since then, adding self-attention into Generative Adversarial Networks has become as standard improvement. [3, 5] demonstrate the superiority of using self-attention in high-fidelity image synthesis models. However, current self-attention mechanisms in GANs only utilize the contextual information within the same image. In this work, we propose a memory cache mechanism that is organized via visual conceptual prototypes. The activation attends not only to its spatial peers for modeling the long range spatial connection within the image itself, but also attends to the corresponding cached memory cells, which allows the attention mechanism to transcend the spatial and temporal limitation.

**Memory Mechanism** Using a memory bank can capture diverse features over a long time period, and has been shown effective in other domains. [36] utilizes the memory bank in contrastive learning in order to obtain more diverse negative samples to contrast with. [10] further improves the stablity of the features accumulated in the memory bank via a momentum-updated encoder. [4] also uses a memory queue during training to accumulate representative negative samples. The Neural Turing Machine [8] introduces an attentional mechanism to allow end-to-end differentiable memory retrieval and overwriting, but they do not use a memory-based pool to modify the activation. SimGAN [26] introduces the image pool trick by using a buffer which stores previously generated samples, in order to make the discriminator focus not only on the current training batch, but also improve itself based on memory.

**Few shot image generation** Few shot image generation task is an challenging task as GAN is very data hungrey and inefficient. Unconditional image generation is especially tricky among these few shot image generation scheme. Different solution has been developed up among the community. Works like DiffAug [40], StyleGAN-Ada [12] propose differentialbe augmentation to avoid discriminator's overfitting. InsGen [37] propose to use Contrastive Learning Objective to enhance the adversial loss in few shot generation setting. We consider these works as proposing better method for discriminators so that discriminators can provide better error signal to the generator. There's no specific requirement of generator architecture in their work. Another line of research lies in proposing generator architectures that can avoid potential mode collapse from the generator

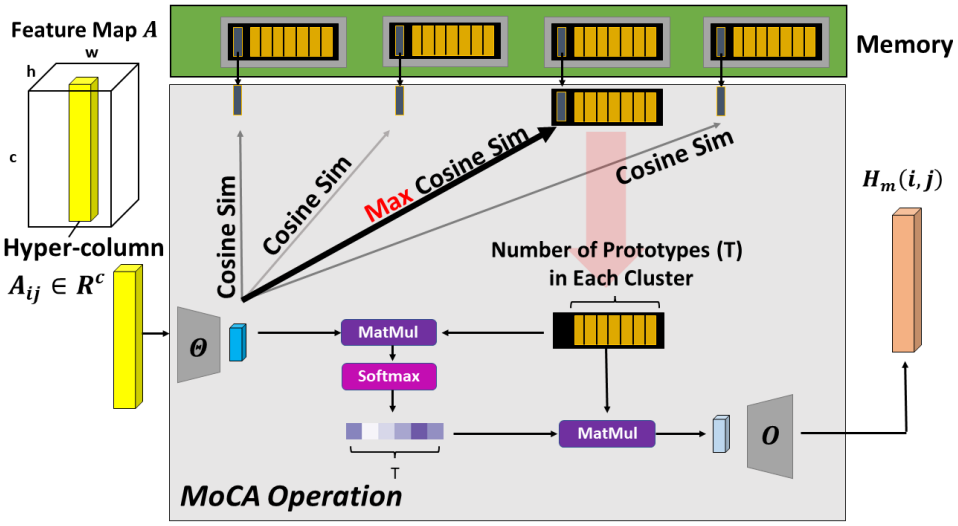



Figure 1: Attention using MoCA. In MoCA, the input activation $A_{ij}$ is first transformed into low dimensional space and used to select its closest prototype semantic cell or "grandmother" cell. After selection, MoCA output the component cells associated with the prototype semantic cell to perform attention. Finally, a network $O(\cdot)$ is used to map the embedding back to the feature space for next layer's propagation.

side. The state-of-the-art architecture for Few shot image generation (hundreds of real images) is FastGAN [19]. Our work also propose a new architectural changes on the generator side and we compare with FastGAN architecture extensively in the experiment section.

## 3 METHOD

Our primary contribution involves a novel prototype-based memory modulation layer to improve the generator network of a GAN. The activation from the previous layer is modified by two attentional processes: 1) attention with Memory Concepts (**MoCA**), and 2) spatial contextual modulation within the generated image itself (**Self-Attention**). The output is a modified feature map that is modulated by these two paths together. We refer a high level overview of our method in Figure 1.

Formally, we denote the input to our MoCA layer to be the activation from the previous layer, $A \in R^{n \times c \times h \times w}$. The output $\hat{A} \in R^{n \times c \times h \times w}$ is the information after contextual modulation. To allow the information being flexibly modulated, we first transform $A$ by 1x1 convolutions into a low-dimensional space via functions $\theta(\cdot)$, $\phi(\cdot)$, and $\psi(\cdot)$, where $\{\theta(A), \phi(A), \psi(A)\} \in R^{n \times \tilde{c} \times h \times w}$.

### 3.1 PROTOTYPE MEMORY ORGANIZATION

In this section, we discuss the organization of our prototype memory that will be used as one route of modulation. Our memory is arranged hierarchically into prototype semantic cells and prototype component cells. As shown in Figure 2, each prototype semantic cell is assigned a set of prototype component cells. Formally, suppose our memory $P$ consists of $M$ prototypes semantics cells $P = \{K_1, K_2, ..., K_M\}$. For each of the prototype semantics cells $K_i$, there are $T$ number of prototype component cells $\{E_1^{(i)}, E_2^{(i)}, E_3^{(i)}, ..., E_T^{(i)}\}$ stored in the memory cell associated with $K_i$, where $E_j^{(i)}$ and $K_i$ both $\in R^{\tilde{c}}$ and $K_i = (\sum_{j=1}^{T} E_j^{(i)})/T$ is the mean of the stored prototype component cells. Those prototype component cells come from the feature maps in previous iterations after being transformed via a momentum updated context encoder $\tilde{\phi}(\cdot)$ and are updated by the memory at the end of every iteration. $\tilde{\phi}(\cdot)$ is a momentum counterpart of $\phi(\cdot)$ and its parameter is updated as shown in Equation 1 with the momentum parameter $m$. The rationale of using the activation of $\tilde{\phi}(\cdot)$ instead of $\phi(\cdot)$ is that it can allow the prototype component cells to contain broader information

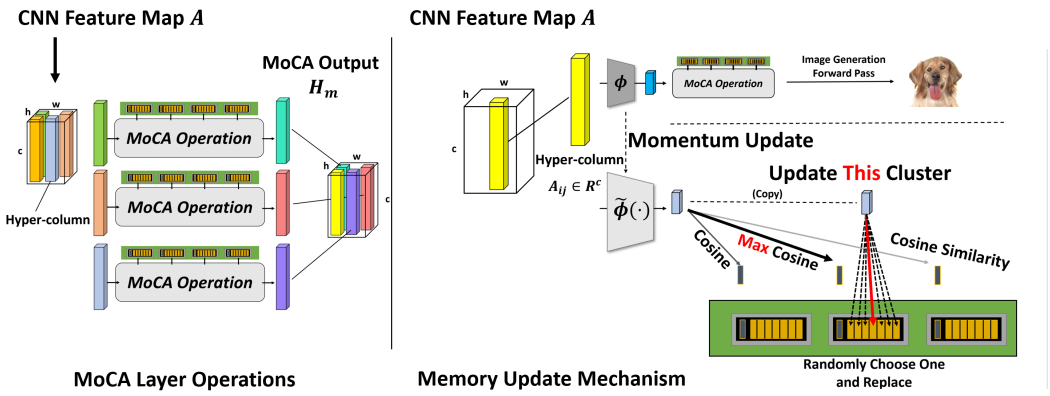



Figure 2: **Left:** MoCA Layer overview. Each hyper-column in the feature map $A$ is processed by the MoCA Operation specified in Figure 1 and then integrated together to pass to the next layer. **Right:** Memory Update Mechanism. When updating the memory, a momentum updated $\tilde{\phi}(\cdot)$ is used to transform the input data into low dimensional space. Then the closest cluster is selected based on cosine similarity. A random update policy is adopted for the selected cluster to avoid trivial solution.

beyond the current training batch, to provide more robust and diverse features to attend to.

$$\tilde{\phi}_\theta \leftarrow \tilde{\phi}_\theta * (1 - m) + \phi_\theta * m \tag{1}$$

After being transformed via $\tilde{\phi}(\cdot)$, each activation column (pixel location) in the feature map is assigned to its closest prototype semantic cell and replaces one of the previous prototype component cells belonging to that semantic pool in the memory. We use a random replacement policy to prevent all prototype component cells collapsing to one trivial solution. The update is done in a batch synchronized fashion, i.e. we update $K_i$ as the mean of its contained prototypes after all prototype component cells are updated for the whole batch.

### 3.2 Memory Concept Attention

As we show in section 3.1, our memory is a cache of many activations gathered from across the whole training set as well as different training periods. By organizing them via semantic prototypes $K_i$, we can route more diverse information into the final activation while still maintaining high data efficiency, hence yielding higher quality in image synthesis task. In this section, we present the detailed operation given an existing memory $P$ and the input feature transformation $\{\theta(A), \Phi(A), \Psi(A)\} \in R^{n \times \tilde{c} \times h \times w}$.

During the memory attentional process, an activation column $\mathbf{a} \in R^{\tilde{c}}$ from $\theta(A)$ first selects the closest prototype $K_i$ and retrieves the associated prototype component cell matrix $\mathbf{E}^{(i)} \in R^{\tilde{c} \times H}$ from memory, where each column $j$ of $\mathbf{E}^{(i)}$ is an prototype component cell $E_j^{(i)} \in R^{\tilde{c}}$. Next step is to attend the activation column $\mathbf{a}$ with the prototype component cell matrix $\mathbf{E}^{(i)}$. First, a similarity score $\mathbf{s}$ is calculated as $\mathbf{s} = [\mathbf{E}^{(i)}]'\mathbf{a}$. A non-linear softmax normalization is then applied to $\mathbf{s}$ to obtain the sparsified attentional weight $\boldsymbol{\beta}$ where for each entry $\boldsymbol{\beta}_t$ ($t = \{1, 2, ..., T\}$) (Equation 2).

$$\boldsymbol{\beta}_r = \frac{\exp(\boldsymbol{s}_r)}{\sum_{i=1}^{T} \exp(\boldsymbol{s}_l)} \tag{2}$$

Using the (softly) sparsified attention weight $\boldsymbol{\beta}$, we can build the retrieved information from memory $\boldsymbol{h}_m = \mathbf{E}^{(i)}\boldsymbol{\beta}$ ($\boldsymbol{h}_m \in R^{\tilde{c}}$) for the activation column $\mathbf{a}$. By applying the same operation to every hypercolumn $\mathbf{a}$ at every spatial location as well as every image, we obtain $\boldsymbol{H}_m \in R^{n \times \tilde{c} \times h \times w}$.

### 3.3 Spatial Contextual Attention

While the memory prior is important, the spatial contextual information also plays a role in information modulation. Therefore, we also use a non-local network to implement the spatial contextual

modulation in the same layer. Specifically, we first compute a affinity map between $\theta(A)$ and $\phi(A)$, denoting as $\boldsymbol{S} = [\theta(A)]^T \phi(A)$. Each row of $\boldsymbol{S}$ is then normalized via softmax to allow a rather sparse attention weight $\hat{\boldsymbol{S}}$ to be computed. $\hat{\boldsymbol{S}}$ is then multiplied with $\phi(A)$ to get spatial contextual modulation $\boldsymbol{H}_s \in R^{n \times \tilde{c} \times h \times w}$.

### 3.4 Integrate two routes of modulation

Finally, we integrate the retrieved information from memory $\mathbf{H}_m$ and the spatial contextual modulation $\boldsymbol{H}_s$ by element-wise addition, $\boldsymbol{H} = \boldsymbol{H}_m \oplus \boldsymbol{H}_s$, and transform $\boldsymbol{H}$ back to the original feature space via 1x1 convolution $\boldsymbol{O}(\cdot)$. A learnable parameter $\gamma$ is multiply to the output of $\boldsymbol{O}(\cdot)$ to adjust its importance before adding a residual connection from the input activation $A$. Overall, the output of our two way modulation $\hat{A} = \gamma \boldsymbol{H} + A$ is output into the next layer in the generator.

## 4 Experiments

Our MoCA module is a generator architecture improvement and is not specific to discriminator's training techniques. Note that some of the existing works on unconditional few shot image generation (DiffAug [40], ADA [12], etc.) improve the quality of image synthesis via focusing on the discriminator side. For example DiffAug [40] propose to perform augmentation of the generated image before fed into the discriminator to prevent discriminator's overfitting. Although this line of works are valuable, they are *orthogonal* and *complementary* to MoCA as MoCA is a generator architecture that can be trained under all of these discriminator training techniques. We perform experiments of MoCA enhanced generator with either DiffAug [40] or ADA [12] augmented discriminators in the following experiments.

**Datasets and Metrics** We validate the performance improvements of adding MoCA layer with 6 datasets of few-shot image synthesis. Namely, we perform compasion between the baseline architecture and the MoCA enhanced architecture on Animal-Face Dog [27], 100-Shot-Obama [40], ImageNet-100 [25], COCO-300 [18], CIFAR10 [16] and Caltech-UCSD Birds (CUB) [35]. Animal-Face Dog consists of 389 dog image while 100-Shot-Obama contains 100 Obama face with different expressions. We use 256x256 resolution for both of these dataset across all experiments. To further challenge the few shot image generation task, we select 100 and 300 images from one class of ImageNet [25] ("Jeep" category) and MSCOCO [18] ("Train" category) respectively to compose ImageNet-100 and COCO-300 dataset (the detail of selected image list can be found in the Appendix 8.2). In addition, we also conduct experiments on CUB and CIFAR10 datasets. CUB is a high resolution bird images in the wild. We only use the training set of CUB to perform generation which results in 5,990 256x256 resolution training images. Besides, we also consider CIFAR10 dataset (60,000 32x32 images span across 10 categories) as it can demonstrate that MoCA can also be helpful under medium scale dataset. For evaluation metrics, we use a widely adopted metric, Fréchet Inception Distance (FID) [11] to measure the quality of generated images. For most of the experiments, except for the evaluation of model robustness against noise, the standard deviation of FID are often less than 1% relatively and much smaller than the performance improvement. To strengthen confidence in the evaluation of the image synthesis performance, we also evaluate the Kernel Inception Distance (KID) metric [2], which is found to be more descriptive on the few shot image generation tasks.

**Baselines** To demonstrate that MoCA is not limited to certain backbone architecture, we install MoCA to two state-of-the-art architectures, FastGAN [19] and StyleGAN2 [13]. Of note, FastGAN is the state-of-the-art architecture that specifically targeted on image synthesis in *extremely* low data regime. It's an important baseline architecture that provide reasonable few shot synthesis given only hundreds of real images with as few as 15 GPU hours of training time. Additionally, we also install MoCA to StyleGAN2, which is more powerful and generic if more resource and time are available. We will showcase that MoCA layer can be useful for generators in both architectures in section 4.1. The implementations of the baseline models are based on their official publicly available PyTorch implementation. We follow the best training configuration reported in the original papers to train our models.

Figure 3: Generated images from MoCA on different datasets. Different resolution of images are considered. The biggest one is 256x256, the middle one is 64x64 and the smallest one on the right is from CIFAR-10 with 32x32 resolution.

### 4.1 Few Shot Image Synthesis Performance

Here, we present the experiments results on 4 small datasets each of which has only hundreds of real images (Animal-Face Dog [27], 100-Shot-Obama [40], ImageNet-100 [25], COCO-300 [18]). Qualitative results are shown in Figure 3. Meanwhile, we show quantitative comparisons between the best performance of MoCA and baseline models for each dataset in Table 1 (FastGAN) and Table 2 (StyleGAN2). Note that all the KID is shown in the scale of $10^{-3}$. As shown in the results, we can observe consistent improvements in terms of image synthesis quality in this difficult low data regime when adding MoCA to the generator. In terms of FID score, we observe that, using FastGAN base architecture , MoCA can bring 5.8% improvement on Animal Face Dog, 13.8% improvement on Obama, 21.7% improvement on ImageNet-100 and 12.4% improvement on COCO-300 dataset when using FastGAN as the baseline models (Table 1). Similar improvements are also observed with KID metrics.

| Generator Architecture | Discriminator Training | Animal Face Dog | | Obama | | ImageNet-100 | | COCO-300 | |
|---|---|---|---|---|---|---|---|---|---|
| | | FID ↓ | KID ↓ | FID ↓ | KID ↓ | FID ↓ | KID ↓ | FID ↓ | KID ↓ |
| FastGAN [19] | DiffAug [40] | 51.25 | 17.11 | 43.13 | 13.52 | 66.44 | 22.31 | 30.43 | 6.02 |
| MoCA-FastGAN (ours) | DiffAug [40] | **48.27** | **13.83** | **37.19** | **7.81** | **52.01** | **7.23** | **26.67** | **4.53** |

Table 1: Quantitative results of MoCA on FastGAN based architecture. All experiements in this table are conducted using 256x256 size images. Results suggest a consistent improvement of MoCA layer across different few shot generation datasets.

| Generator Architecture | Discriminator Training | Animal Face Dog | | Obama | | ImageNet-100* | | COCO-300* | |
|---|---|---|---|---|---|---|---|---|---|
| | | FID ↓ | KID ↓ | FID ↓ | KID ↓ | FID ↓ | KID ↓ | FID ↓ | KID ↓ |
| StyleGAN2 [13] | ADA [12] | 58.30 | 22.72 | 28.22 | 5.44 | 54.32 | 4.53 | 77.81 | 38.66 |
| MoCA-StyleGAN2 (ours) | ADA [12] | **55.35** | **17.62** | **25.93** | **4.91** | **46.71** | **3.03** | **64.32** | **20.53** |

Table 2: Quantitative results of MoCA on StyleGAN2 based architecture. We use 64x64 image size when performing experiments on StyleGAN2 and MoCA-StyleGAN2 on ImageNet-100* and COCO-300* and 256x256 image size synthesis on Animal Face Dog and Obama dataset. Consistent improvements when adding MoCA layer are observed despite the image size and datasets.

To further test MoCA's enhancement on more powerful model, we run experiments with StyleGAN2 base architecture. Using FID evaluation metrics, we observe 5.1% improvement on Animal Face Dog dataset, 8.1% improvement on Obama dataset, 14.1% improvement on ImageNet-100 dataset and 17.3% improvement on COCO-300 dataset. Note that since ImageNet-100 and COCO-300 dataset are very challenging as it contain much diverse scenes than Animal Face Dog and Obama. To compensate the model size of StyleGAN2 base architecture, we downsize ImageNet-100 and COCO-300 to 64x64 image size when training using StyleGAN2 and MoCA-StyleGAN2. From the results we observe that MoCA-StyleGAN2 consistently synthesizes better images comparing to the

| Generator Architecture | CIFAR-10 | |
|---|---|---|
| | FID ↓ | KID ↓ |
| StyleGAN2 [13] | 5.49 | 2.43 |
| SA-StyleGAN2 [38] | 5.60 | 2.79 |
| MoCA-StyleGAN2 (ours) | **4.68** | **1.39** |

| Generator Architecture | Animal Face Dog | | Obama | | ImageNet-100 | | COCO-300 | |
|---|---|---|---|---|---|---|---|---|
| | FID ↓ | KID ↓ | FID ↓ | KID ↓ | FID ↓ | KID ↓ | FID ↓ | KID ↓ |
| FastGAN [19] | 51.25 | 17.11 | 43.13 | 13.52 | 66.44 | 22.31 | 30.43 | 6.02 |
| SA-FastGAN [38] | 51.17 | 16.57 | 38.93 | 9.37 | 56.69 | 16.93 | 29.66 | 6.31 |
| MoCA-FastGAN (ours) | **48.27** | **13.83** | **37.19** | **7.81** | **52.01** | **7.23** | **26.67** | **4.53** |

Table 3: Ablation study on the improvement of MoCA comparing with Self-Attention [38]

baseline despite the image size. Additionally, we also observe that although different augmentation methods on the discriminator side don't affect the MoCA's improvement, as FastGAN based models use DiffAug whereas StyleGAN2 based models employ ADA technique.

### 4.2 Ablation Studies

**Self-Attention v.s. MoCA** On both StyleGAN2 and FastGAN, we compare the performance of MoCA with a standard self-attention module to verify that the performance improvement of our model is not only due to the self-attention block, as several previous works have highlighted the advantage of applying self-attention on GANs to capture non-local dependency [3, 39]. Table 3 shows that in spite of the baseline architecture, MoCA consistently outperform Self-Attention module, suggesting that MoCA module plays an irreplaceable role in our method.

| # Cluster | Size of Cluster | FID ↓ | KID ↓ |
|---|---|---|---|
| 1 | 512 | 5.30 | 2.53 |
| 1 | 8192 | 4.76 | 1.52 |
| 3 | 256 | 5.10 | 1.96 |
| 20 | 256 | 4.91 | 1.61 |
| 20 | 1024 | 4.90 | 1.56 |
| 32 | 256 | **4.68** | **1.39** |

Figure 4: StyleGAN2 on CIFAR-10 with different architectures. The first column denotes the number of cluster in the memory, 1 indicates no clustering mechanism is applied. The second column denotes the number of concept vectors attributed to a single cluster.

**Importance of memory clustering organization** The second aspect we study is the impact of the clustering mechanism on the overall model performance. We find that for MoCA, a single concept pool with sufficiently large size, can still has similar performance compared against MoCA with multiple clusters. As shown in Table 4, the performance of no-cluster MoCA containing 8192 concepts inside, is at the similar level as MoCA with 32 clusters. We hypothesize this is because, although there is no clustering mechanism to enforce the memory bank learning diverse concepts in no-cluster MoCA, given a very large pool of concepts, it is still able to memorize diverse and useful concepts. Despite the finding that no-cluster MoCA can have similar performance to cluster MoCA with a sufficiently large pool, we argue that using the clustering mechanism is still superior for two reasons. First, it is difficult to identify an appropriate pool size for no-cluster MoCA to have performance equivalent to cluster MoCA. Second, using the clustering mechanism greatly reduces the amount of vectors involved in the attention calculation, which enables us to build a larger memory bank for complex datasets.

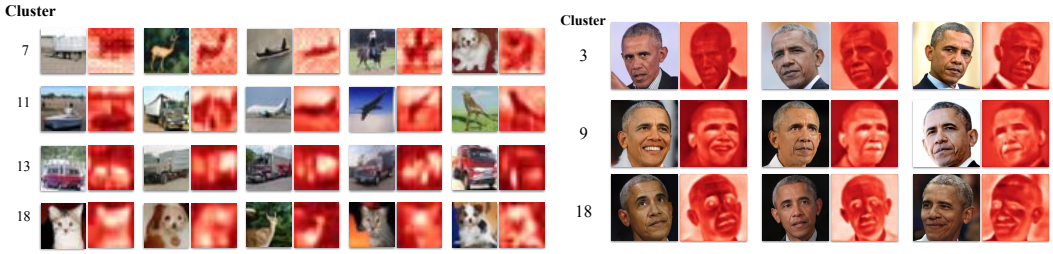
**Importance of Momentum update mechanism** We also study the effect of different ways to update the concept encoder. Table 4 shows that the models' FID on multiple datasets generally improves when using momentum to update the concept encoder. With momentum, the FID of FastGAN on larger and diverse dataset like AnimalFace Dog and CUB are notably stronger than model without momentum. This supports our motivation for using a momentum-updated encoder to build a more generalized memory bank that contains broader information beyond the current training batch.

| Model | FID↓ | | | |
|---|---|---|---|---|
| | AnimalFace Dog | Obama | Grumpy-cat | CUB |
| MoCA with momentum | **48.27** | **37.19** | **25.68** | **25.66** |
| MoCA *w/o* momentum | 63.08 | 46.25 | 29.49 | 51.74 |

Table 4: Ablation study on FastGAN + MoCA with different update mechanism

### 4.3 Prototype Concept Analysis

Human brains are few shot learning systems that can organize our past experience into various concepts in our cognitive mind. Those concepts are the corner stones that support our actions, thoughts and feelings. It is suspected that the grandmother cells in the brain provide certain ability




(a) CIFAR-10 (b) 100-shot Obama

Figure 5: Affinity heat-map w.r.t. different concept clusters. White: strongest affinity; Dark red: zero affinity. **Left (a):** MoCA-StyleGAN2 on CIFAR10. **Right (b):** MoCA-FastGAN on 100-shot Obama dataset. We can observe that Different cluster associated with different semantic meanings although not trained with category supervision.

related to discrete concepts formation. As inspired by the neuroscience property of "grandmother" cells, we demonstrate that MoCA can unsupervisely learn distinct semantic concepts during the analysis-by-synthesis path. To illustrate the meaning behind each semantic concept, we take two different approaches: **(1)** we demonstrate how the intermediate representations during synthesis are viewed from the semantic concepts' perspective. **(2)** Since it's a generation process, we'd like to "activate" certain concept in MoCA and observe the generation results.

**A Semantic Concept's Perspective** As MoCA is a memory layer that modulate the image generation process via attention, one important aspect to understand MoCA's learnt concepts is by finding out the image patterns the concepts are associated with. For each cluster in MoCA's memory, we consider the cosine similarity between its mean vector and the intermediate feature map of the layer MoCA is installed upon. This will result in a similarity score for each spatial location, hence a spatial affinity map will be generated for each cluster during every generation. This affinity map will highlight the region in the generated feature map that is similar to the cached vector cluster (concept). We can then understand this affinity map as an approximation about where is this concept (cluster) located in the generated images. In other words, it provide a sense of what image pattern this concept is associated with.

In Figure 5a and Figure 5b, we show the resulting affinity maps computed as discussed above (detailed schematic of visualization procedure is shown in Appendix Figure 7). Note that we select clusters and images for demonstration purpose. Results on CIFAR10 using MoCA-StyleGAN (32x32 resolution) and Obama dataset using MoCA-FastGAN (256x256 resolution) are acquired. We find that these affinity maps demonstrate that concepts formed during MoCA training cover crucial and diverse patterns for image synthesis. As shown in Figure 5a and Figure 5b, patterns of different clusters are usually semantic meaningful and very distinct. On CIFAR-10, we notice that cluster 11 stores concepts related to the sky, while cluster 13 stores concepts related to the frame of trucks. Cluster 7 favors the white space while cluster 18 has a "crush" on animals' body and head. Similarly, on the other dataset, different clusters also focus on very different patterns. For instance, on 100-shot Obama, cluster 3 stores the concepts closely related to ties, and cluster 9 contains concepts related to the light reflection on the face.

(For more analysis and interpretation detail, we refer readers to the Section 8.5 in the Appendix)

**"Activating" Semantic Concepts** Another method towards understanding what MoCA has been learning is by "activating" the semantic concepts of MoCA during the generation. Because prototypes in MoCA's memory only affect the generation process via attentive process, we can't generate images only based on the cluster mean without proper activation from previous layers. To find an approximation, we search for the intermediate hyper-column in the feature map that is the most similar to the cluster mean vector when generating each image. After search, we use the same input noise vector to perform the generation pass again except this time we zero-out *all other* hyper-columns' activation in the feature map in the MoCA layer. This leads to partial image generation as shown on the left of every image in Figure 6. Interestingly, "activating" different clusters using this method leads to generation that is associated with semantically meaningful sub-parts (e.g. sky, lights, etc.). In Figure 6, we show the representative generation results on COCO-300 dataset with MoCA-FastGAN architecture (256x256 resolution). Each column in the figure displays the generation when "activating" one cluster with different generation noise input as each row entry. From the results we can clearly observe that different clusters are semantically different as the closest activation in the feature maps lead to distinctive parts generation, suggesting meaningful semantics have unsupervisely

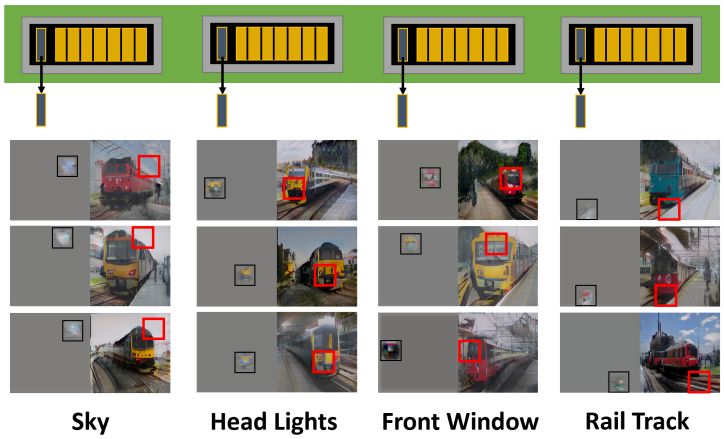



Figure 6: Visual Concepts visualization.Generation results when only activating the closest hypercolumun to the cluster mean vectors. We observe that clusters in MoCA learn distinct semantics by the analysis-by-synthesis path, without human supervision.

emerged in MoCA. Note that the label under each column of Figure 6 is added by using human commonsense and merely serve to guide the understanding of the semantics of each cluster.

### 4.4 Robustness against noise

We evaluate the noise robustness of our proposed memory concept attention and compare it against the standard self-attention module. The evaluation is conducted on FastGAN models trained without any injection of noise into the intermediate layers. We select FastGAN because we find that its performance varies little with or without injecting noise during training, which enables us to factor out the influence of noise injected during training. In the evaluation stage, we inject Gaussian noise of different magnitudes with variance from 0 to 1 into the input feature map of the MoCA layer and standard self-attention layer. Our results (Table 5) show that, without the help of external memory, self-attention is more vulnerable to the noise attack, while MoCA is generally less sensitive to noise. In MoCA, the feature map can attend to previously-stored concepts in the memory bank to retrieve noise-free information, which alleviates the influence of noise perturbation. We also find that, for no-cluster MoCA, despite inferior performance on noise-free data, is the most robust model under higher levels of noise. We hypothesize that this is because the feature map can attend to more concepts in no-cluster MoCA, which increases the chances of pixels in the feature map retrieving the appropriate concepts.

| Model | Noise level | | | | |
|---|---|---|---|---|---|
| | 1 | 0.7 | 0.5 | 0.3 | no noise |
| Self-attention | 163.71 (±7.68) | 114.17 (±5.10) | 75.48 (±4.32) | 45.42 (±3.61) | 26.65 (±0.33) |
| MoCA (*w/o cluster*) | **79.71**(±2.79) | **63.84** (±1.68) | **54.63** (±1.02) | 45.82 (±1.28) | 37.04 (±0.31) |
| MoCA (*with cluster*) | 117.38 (±4.32) | 83.86 (±4.40) | 59.24 (±2.63) | **38.93** (±3.71) | **23.98** (±0.12) |

Table 5: Comparison of models' FID on CUB dataset under different levels of noise injection.

## 5 Conclusion

Our study is triggered by the question of the computational advantages of having prototype-like "grandmother" neurons in the early visual cortex. Our experiments demonstrate some significant benefits of the MoCA mechanism, particularly in few-shot image generation. Our work establishes the viability of one hypothesis for the potential computational advantages of the extremely sparse, diverse and complex feature detectors in V1. The MoCA mechanism introduced might be useful as a variant of the self-attention mechanism, particularly for few-shot learning. Additionally, in this study, we focus primarily on the top-down image synthesis path, but we envision that the MoCA mechanism could also be useful for the analysis path as in deep CNNs with self-attention, or in the interaction between the two paths for modeling more fully the function of the visual system.

## 6 ETHICS STATEMENT

This study investigates a novel proposal for understanding the computational advantages of having prototype-like “grandmother” neurons in the visual cortex, namely, that these prototype neurons can serve as auxiliary memory priors to modify both the embedding space of the self-attention mechanism as well as the activation patterns of neurons during the image synthesis process. While the motivation is for advancing our understanding of contextual modulation in neural systems, our work also results in a novel mechanism for contextual modulation in deep learning systems, particularly in image generation, and thus has a broader positive impact on machine learning, particularly on few-shot learning. However, image generation technology can be misused for misinformation, with adverse social impact, e.g. the creation of "deepfakes." On the other hand, advancing our understanding of image generation is also crucial for combating deepfakes and other misuses of image generation technology by bad actors.

## 7 REPRODUCIBILITY STATEMENT

We provide an anonymous source code link for reproducing our result in the supplementary material and include complete files which can reproduce the data processing steps for each dataset we use in the supplementary material.

## REFERENCES


[1] Elie Bienenstock, Stuart Geman, and Daniel Potter. Compositionality, mdl priors, and object recognition. *Advances in neural information processing systems*, pages 838–844, 1997.

[2] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

[5] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *arXiv preprint arXiv:2012.09841*, 2020.

[6] Stuart Geman, Daniel F Potter, and Zhiyi Chi. Composition systems. *Quarterly of Applied Mathematics*, 60(4):707–736, 2002.

[7] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[8] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[9] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[12] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data, 2020.

[13] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[15] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[17] Tai Sing Lee and David Mumford. Hierarchical bayesian inference in the visual cortex. *JOSA A*, 20(7):1434–1448, 2003.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[19] Bingchen Liu, Yizhe Zhu, Kunpeng Song, and Ahmed Elgammal. Towards faster and stabilized gan training for high-fidelity few-shot image synthesis. *arXiv e-prints*, pages arXiv–2101, 2021.

[20] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2017.

[21] James L McClelland and David E Rumelhart. Distributed memory and the representation of general and specific information. In *Connectionist psychology: A text with readings*, pages 75–106. Psychology Press, 2020.

[22] David Mumford. On the computational architecture of the neocortex. *Biological cybernetics*, 66(3):241–251, 1992.

[23] R Quian Quiroga, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–1107, 2005.

[24] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[26] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017.

[27] Zhangzhang Si and Song-Chun Zhu. Learning hybrid image templates (hit) by information projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1354–1367, 2012. doi: 10.1109/TPAMI.2011.227.

[28] Shiming Tang, Tai Sing Lee, Ming Li, Yimeng Zhang, Liu Fang Xu, Y, Teo Benjamin, and Hongfei Jiang. Complex pattern selectivity in macaque primary visual cortex revealed by large-scale two-photon imaging. *Current Biology*, 28:38–48, 2018.

[29] Shiming Tang, Yimeng Zhang, Zhihao Li, Ming Li, Fang Liu, Hongfei Jiang, and Tai Sing Lee. Large-scale two-photon imaging revealed super-sparse population codes in the v1 superficial layer of awake monkeys. *elife*, 2018.

[30] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[32] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

[33] Jianyu Wang, Zhishuai Zhang, Cihang Xie, Yuyin Zhou, Vittal Premachandran, Jun Zhu, Lingxi Xie, and Alan Yuille. Visual concepts and compositional voting, 2017.

[34] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[35] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[36] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. *arXiv preprint arXiv:1805.01978*, 2018.

[37] Ceyuan Yang, Yujun Shen, Yinghao Xu, and Bolei Zhou. Data-efficient instance generation from instance discrimination. *arXiv preprint arXiv:2106.04566*, 2021.

[38] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[39] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks, 2019.

[40] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training, 2020.

[41] Song-Chun Zhu and David Mumford. *A stochastic grammar of images*. Now Publishers Inc, 2007.

# 8 APPENDIX

## 8.1 ALGORITHM

Here, we present the algorithm that is presented in Figure 1 and Figure 2 in the main text.

**Algorithm 1:** MoCA + Self-Attention in the Generator

**Result:** Updated activation $\hat{A}$ after MoCA and Self-Attention; Updated Memory $P$.
$A \in R^{n \times c \times h \times w} \leftarrow$ from previous layer;
$\{\theta(A), \phi(A), \psi(A)\} \in R^{n \times \tilde{c} \times h \times w} \leftarrow$ transform into $\tilde{c}$ dimensional space via 1x1 conv;
# Memory Concept Attention;
**for** *every spatial column* $\boldsymbol{a} \in R^{\tilde{c}}$ *in* $\theta(A)$ **do**
  Choose prototype semantic cell $K_i$ where $i \leftarrow \text{argmin}_i ||K_i - \mathbf{a}||_2^2$;
  $\mathbf{E}^{(i)} \in R^{\tilde{c} \times h} \leftarrow$ Retrieve Prototype Component Cells for $K_i$ from memory;
  $\boldsymbol{s} \leftarrow [\mathbf{E}^{(i)}]' \mathbf{a}$ # compute dot product with memory;
  $\boldsymbol{\beta} \leftarrow \frac{exp\{\boldsymbol{s}\}}{\sum_{l=1}^{T} exp\{\boldsymbol{s_l}\}}$;
  $\boldsymbol{h}_m \leftarrow \mathbf{E}^{(i)} \boldsymbol{\beta}$ # $\boldsymbol{h}_m \in R^{\tilde{c}}$;
**end**
$\mathbf{H}_m \leftarrow$ Combining $\boldsymbol{h}_m$ # $\mathbf{H}_m \in R^{n \times \tilde{c} \times h \times w}$;
# Self-Attention;
$\boldsymbol{S} \leftarrow [\theta(A)]^T \phi(A)$;
$\hat{\boldsymbol{S}} \leftarrow softmax(\boldsymbol{S})$;
$\boldsymbol{H}_s \leftarrow \psi(A) \hat{\boldsymbol{S}}$ # $\mathbf{H}_s \in R^{n \times \tilde{c} \times h \times w}$;
$\boldsymbol{H} \leftarrow \boldsymbol{H}_m \oplus \boldsymbol{H}_s$;
$\hat{A} = \gamma O(\boldsymbol{H}) + A$;
# Memory Update;
$A_m \leftarrow \tilde{\phi}(A)$;
**for** *every column* $\boldsymbol{a}_m \in R^{\tilde{c}}$ *in* $A_m$ **do**
  Choose prototype semantic cell $K_i$ where $i \leftarrow \text{argmin}_i ||K_i - \mathbf{a}_m||_2^2$;
  Randomly update one column in $\mathbf{E}^{(i)}$ with $\mathbf{a}_m$;
**end**
$\tilde{\phi}_\theta \leftarrow \tilde{\phi}_\theta * (1 - m) + \phi_\theta * m$ # momentumly update the $\tilde{\phi}$

## 8.2 EXPERIMENTAL DETAILS

In this section we provide details for how we implement MoCA on FastGAN and StyleGAN 2 respectively. For reference, we provide our code and image ID list of MSCOCO, ImageNet subset used in the experiments at https://anonymous.4open.science/r/MoCA-D372/README.md. All the datasets used in experiments are publicly available and for non-commercial usage only. Among them, CIFAR-10[1] is attributed to [15], Obama, Animal Face, Grumpy Cat[2] are attributed to [40], and Caltech-UCSD Birds-200-2011[3] (CUB) is attributed to [32].

We insert MoCA layer between the last but one residual block and the last residual block of the generator. Given the input feature map $\mathbf{A}$ of the last but one residual block $f^{(l-1)}$, the last residual block $f^{(l)}$ and MoCA module $\Phi$, suppose the original image generation procedure on the last two blocks is:

$$\hat{\mathbf{x}} = f^{(l)}\left(f^{(l-1)}(\mathbf{A})\right). \tag{3}$$

Then the feedforward procedure including MoCA can be defined as:

$$\hat{\mathbf{x}} = f^{(l)}\left(\Phi\left(f^{(l-1)}(\mathbf{A})\right)\right), \tag{4}$$

where $\hat{\mathbf{x}}$ is the generated image. Please note that, to enable that MoCA can be flexibly integrate into arbitrary layer, MoCA will not change the channel numbers and resolution of the feature map.

[1] https://www.cs.toronto.edu/~kriz/cifar.html
[2] https://hanlab.mit.edu/projects/data-efficient-gans/datasets/
[3] http://www.vision.caltech.edu/visipedia/CUB-200-2011.html

#### 8.2.1 FastGAN

**Network Architecture and Optimization** We use exactly the Fast-GAN architecture Liu et al. [19] as the backbone. MoCA layer is inserted after the 64x64 resolution layer for all experiments run for CUB, AnimalFace Dog, Obama and Grumpy-cat. We use Adam optimizer for training and 0.0002 for learning rate with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Differentialbe augmentation is used during training. We utilize the loss proposed in [19] for stable and optimal training. For all our experiments, 100K iteration is run to perform fair comparsion.

**Computational Resource** With FastGAN and MoCA, it takes about 15 hours to train 100K iteration on a single GTX-1080Ti GPU.

#### 8.2.2 StyleGAN 2

**Network Architecture and Optimization** We use exactly same architecture from [12] as the backbone. We use Adam [14] optimizer with a learning rate of $0.0025$ and $\beta_1 = 0, \beta_2 = 0.99$. We adopt the same training configuration as the Official PyTorch implementation of StyleGAN 2 (including Adaptive discriminator augmentation, path length regularization , lazy regularization [13], and non-saturating logistic loss [7]). For the training on the large dataset (CIFAR-10), we train models for 200k iterations of gradient updates, and 100k iterations for few-shot datasets.

**Computational Resource** We train StyleGAN2-based model with 4 RTX-2080Ti GPUs each run. The training of each model takes roughly 20 - 35 hours to complete, depending on the image resolution.

### 8.3 MoCA on LS-GAN

To strengthen our conclusion on MoCA, we have also conducted experiments on the least-square GAN (LS-GAN)[20] framework which has a different loss and architecture compared to FastGAN and StyleGAN. We found MoCA can still provide a certain amount of improvement, consolidating the advantages of the MoCA layer.

| Generator Architecture | Discriminator Augmentation | Animal Face Dog | | Obama | | Cifar-10 | |
|---|---|---|---|---|---|---|---|
| | | FID ↓ | KID ↓ | FID ↓ | KID ↓ | FID ↓ | KID ↓ |
| LS-GAN [20] | DiffAug [40] | 135.83 | 109.32 | 180.35 | 226.45 | 56.37 | 4.21 |
| MoCA-LS-GAN (ours) | DiffAug [40] | **122.89** | **91.14** | **172.97** | **206.38** | **50.48** | **3.86** |

Table 6: Quantitative results of MoCA on LS-GAN based architecture. All experiements in this table are conducted using 64x64 size images. Results suggest a consistent improvement of MoCA layer.

### 8.4 Limitation and failure

Despite MoCA can bring in consistent improvement on most of the dataset with limited data, we also observe case where MoCA is less effective. We found that adding MoCA on StyleGAN for training Grumpy-cat leads to a slight performance decrease and leads to small improvement on FastGAN. This suggests that when the underlying dataset is less diverse and the base network is quite large, MoCA cached concepts can potentially cause more distraction during generation, leading to performance setbacks.

| Generator Architecture | Discriminator Augmentation | Grumpy Cat | |
|---|---|---|---|
| | | FID ↓ | KID ↓ |
| FastGAN [19] | DiffAug [40] | 26.65 | 5.71 |
| MoCA-FastGAN (ours) | DiffAug [40] | **25.68** | **5.18** |
| StyleGAN2 [13] | ADA [12] | **24.17** | **3.82** |
| MoCA-StyleGAN2 (ours) | ADA [12] | 24.65 | 3.98 |
| LS-GAN [20] | DiffAug [40] | 99.92 | 88.43 |
| MoCA-LS-GAN (ours) | DiffAug [40] | **83.82** | **73.22** |

Table 7: Quantitative results of MoCA on Grumpy-cat dataset.

### 8.5 MEMORY CONCEPT ANALYSIS

We generate 50,000 images from random noise for CIFAR-10, and 1,000 images for 100-shot Obama. For each feature map and specific concept cluster $i$, we count how many pixels' closest cluster (measured by the magnitude of the dot product) is $i$, and repeat this process for all clusters. Then we sum the count and pick top-K clusters which have the most pixels, and visualize their affinity heatmap. To interpret the vector embeddings (concepts) stored in each concept cluster, we perform three kinds of visualization: affinity heatmap w.r.t. different concept clusters, images' binary mask w.r.t. different concept clusters, and t-SNE visualization [30] of the feature map before and after attending with memory concept. The detailed information retrieval process is shown in Figure 7. We use StyleGAN2+MoCA's results for visualization, as they perform the best on most of the dataset.

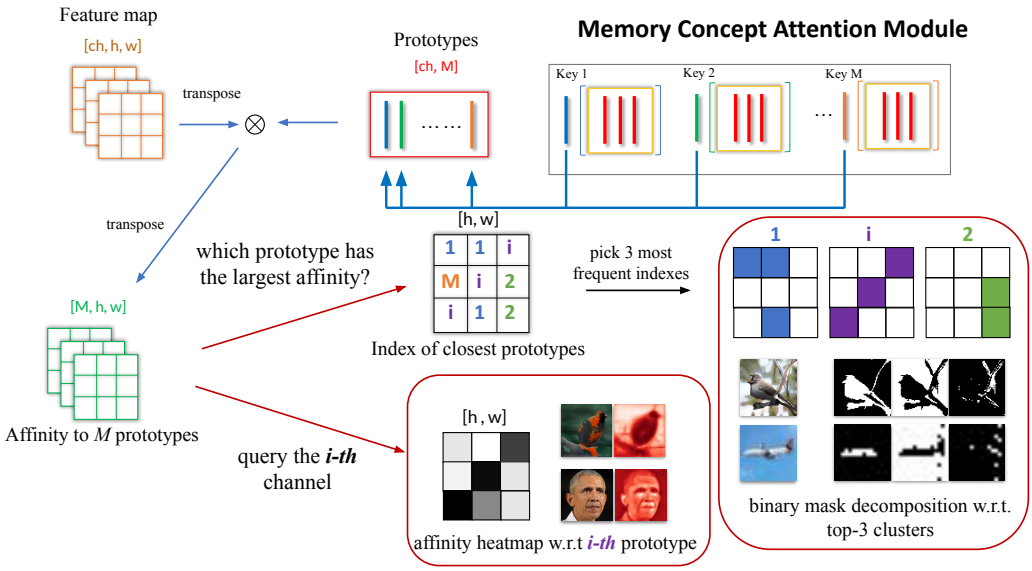



Figure 7: Schematic of affinity map visualization procedure

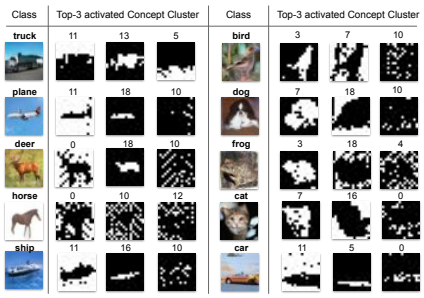
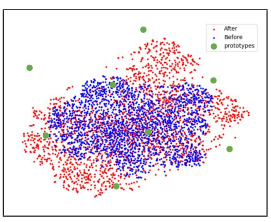
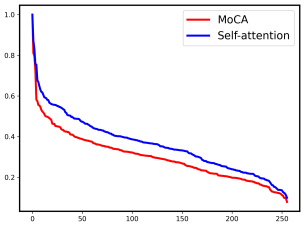



Figure 8: CIFAR-10 image binary decomposition w.r.t. their top-3 activated clusters.



(a) t-SNE visualization of MoCA dynamics




(b) Sharpening of the Context Attention Map by MoCA

Figure 9: Modification of the activation representation and Context Attention Map by MoCA.

**Prototype attention** We study the internal dynamics of MoCA by using t-SNE to visualize the changing trend of feature map before and after attending with concepts in the memory. We generate 10,000 images from random noise on CIFAR-10 and extract the feature maps before and after MoCA layer. Figure 9a shows that the activation patterns (blue dots) from previous layer, after the $\theta$ transform, are effectively influenced by MoCA (green box in Figure 1) into a representation (red dots) that are pulled towards or stretched by certain familiar prototype exemplars (green dots).

**Image synthesis as concepts assembling** By visualizing the affinity to the prototype of each pixel in the feature map, we acquire a high-level understanding of what concepts different clusters are trying to capture. We can also further investigate the relationship between different concepts and

the image synthesis process. We decompose the image into different binary masks with regard to top-3 clusters. The decomposition is shown in Figure 8. Concretely, given a specific cluster $C_k$ and a feature map $F$, we generate the binary mask w.r.t. $C_k$ by querying whether pixels'($F_{ij}$) closest cluster (highest affinity) is $C_k$ or not, where white denotes true and black denotes false. We pick the top-3 clusters for each image and visualize their binary mask as shown in Figure 8. For most images, concepts from the top-2 clusters are usually related to foreground and background information , while the third cluster contains concepts related to high-frequency details in an image. This demonstrate that, with MoCA, image synthesis can also be viewed as a process of retrieving different concepts from memory and assembling them together to create a realistic image.

**Implicit influence of MoCA the attention mechanism** Interestingly, we also observe that having the MoCA module can in turn sharpen the functional activities of the horizontal interactions. To visualize the functional connectivity of the horizontal connections, we plot the normalized rank-order affinity score of the self-attention map in the 16x16 resolution layer of the converged StyleGAN2 trained on CIFAR10.
In Figure 9b, the purple curve represents the rank-ordered curve without installing the MoCA and the red represents that with MoCA. Clearly, MoCA sharpens the functional connectivity among the horizontal connections, even though we do not explicitly modify the self-attention process in the MoCA layer, indicating that our prototype memory is implicitly influencing the horizontal connections.

### 8.6 Metric evaluation

For all models, we evaluate metric using the last 3 snapshots of models (with an interval of 10k gradient updates) and report the best one. We train all StyleGAN-based models for 200k iterations and all FastGAN-based models for 100k iterations. To measure the model performance on Animal-Face Dog [27], 100-Shot-Obama [40], ImageNet-100 [25], COCO-300 [18], we sample $3,000$ noise vectors from standard Gaussian and calculate FID and KID score of the generated images. On CIFAR10 [16] and Caltech-UCSD [35], we sample $5,0000$ respectively to adjust for number of training data.