Supplementary Material

6. Extended Related Work

6.1. Visual Prompting

Visual Prompting (Bar et al., 2022; Bahng et al., 2022; Jia et al., 2022; Xu et al., 2023; Zhang et al., 2024b; Bai et al., 2023) is a class of approaches to adapt computer vision models to downstream tasks, inspired by the success of prompting in NLP (Brown et al., 2020). Approaches like (Bahng et al., 2022; Jia et al., 2022) seek to improve task-specific performance by adding trainable prompt vectors to the model. Other Visual Prompting approaches allow a model to handle various vision tasks (Bar et al., 2022; Xu et al., 2023; Bai et al., 2023; Zhang et al., 2024b) by introducing visual examples or text at the time of inference. Such prompting is related to the way in-context learning (Xie et al., 2021; Wei et al., 2022; Liu et al., 2021; Lu et al., 2021) operates in language models (Radford et al., 2019; Brown et al., 2020; Wang & Komatsuzaki, 2021). In fact, trainable prompts and in-context learning can be viewed as two complementary approaches for "describing" a task to a model (Li & Liang, 2021). Our goal here is to better understand the underlying mechanism of Visual ICL, and we analyze the MAE-VQGAN model presented in (Bar et al., 2022).

6.2. Explainability

In the context of enhancing model interpretability (Zhang et al., 2021; Moraffah et al., 2020; Singh et al., 2024) within computer vision, the integration of Causal Interventions (Bau et al., 2018; Park et al., 2023) and Activation Patching (Zhang & Nanda, 2023) has become a cornerstone in the elucidation of complex neural networks' decision-making processes. These methodologies enable a systematic examination of how models encode and utilize high-level concepts, offering profound insights into their internal mechanisms(Zhang et al., 2024a; Wu & Varshney, 2024; Lu et al., 2023). Causal Interventions (Pearl, 2022; Meng et al., 2022) facilitate the exploration of the causal structures of models by manipulating their internal states or inputs (Gandelsman et al., 2023) and observing the impact on outputs, thus uncovering the direct and indirect effects that drive model predictions. Here we use Activation Patching (Zhang & Nanda, 2023) to show through targeted interventions the significance of Task Vectors towards guiding Visual Prompting models to perform different computer vision tasks.

6.3. Task Vectors

In (Hendel et al., 2023; Todd et al., 2023; Ferry et al., 2023), a Task Vector or Function Vector is a latent activation derived from a particular layer of the transformer (Vaswani et al., 2017). This vector subsequently substitutes the original latent states at the same layer during the forward pass of a query to guide the model to perform the desired task. The investigation of Task Vectors aligns with broader efforts in the field to make neural networks more adaptable and tailored to specific tasks(Liu et al., 2023; Luo & Specia, 2024) as well as boosting the performance(Palit et al., 2023; Xu et al., 2024; Jin et al., 2024) by gaining a deeper understanding of how different layers within a model contribute to its overall function. There have been other similar efforts.

7. Experiments and Results

7.1. Activation Scoring Analysis

Here our goal is to evaluate in isolation the Activation Scoring step (outlined in Section 2.1), specifically whether high scoring activations indeed correspond to Task Vectors.

Collecting Activations. As outlined in Section 2.1, our first step towards computing activation scores is to run the forward pass of the model in a one-shot setting to collect activations across different tasks. Specifically we use 100 prompts and queries from Pascal 5i (Shaban et al., 2017) training set, ensuring that the one-shot performance across all tasks works reasonably well. During the forward pass, we save the activations for every task j and $i = (l, m, k)$, which corresponds to the intermediate activation of the k^{th} token in the l^{th} block after the m^{th} attention head. Afterwards, we compute the mean activation $\mu_{i,j}$ and score $\rho_{token}(i)$.

Evaluation via Clustering. Next, we wish to analyze if $\rho_{token}(i)$ indeed captures "taskness". Intuitively, we expect layers that capture task information to succeed in clustering activations by task. To assess this, we analyze the clustering

Table 2. **Task Clustering Quality.** Clustering Scores of Different Attention Heads, ranked by our Activation Scoring metric (see Section 2). This indicates that higher Activation Scores indeed correlate with better clustering by tasks.

	(Layer, Head)	Our Score ↑	Silhouette Score ↑	Davies-Bouldin Score ↓
High 1	(26, 3)	**2.1663**	**0.3583**	**1.2744**
High 2	(11, 3)	1.0827	0.2692	1.5567
High 3	(11, 13)	0.9849	0.2246	1.8084
High 4	(22, 3)	0.9448	0.2031	2.6842
Random 1	(4, 3)	0.2329	0.0708	4.1062
Random 2	(18, 16)	0.1259	0.0369	4.3256
Random 3	(32, 6)	0.3253	0.0827	2.9670
Random 4	(11, 14)	0.1196	0.0409	5.3884
Random 5	(18, 16)	0.1259	0.0369	4.3256
Low 1	(2, 16)	0.0221	-0.0518	21.8265
Low 2	(2, 12)	0.0264	-0.0334	13.5982
Low 3	(27, 5)	0.0281	-0.0280	15.4219

performance of vectors high ranking activations versus those marked with low scores. We measure the clustering performance using common clustering metrics like the Silhouette Score (Rousseeuw, 1987) and the Davies-Bouldin Score (Davies & Bouldin, 1979). Finally, we also perform a qualitative analysis by visualizing the representations on a t-SNE(van der Maaten & Hinton, 2008) plot, coloring each data point by it's task label.

7.2. Activation Scoring Analysis

We compute $\rho_{token}(i)$ and display it aggregated per head on a heatmap where the y-axis is the layer and the x-axis is the attention head index (see Figure 2). This heatmap showcases certain heads that stand out, suggesting that these heads may hold task vectors. From this heatmap we then select the top two heads ranked by score which are at position $(26, 3)$ and $(11, 3)$; we also select a lower-ranked head at position $(27, 5)$. For each of these three heads, we visualize the clustering of its activations, and the individual Activation Score per token. Both are placed to the right side of the heatmap.

Clustering Visualization. We observe a clear clustering based on task in heads ranked highly–head (26,3) for instance–by our scoring methodology, and what appears to be many small clusters for the low-ranked head which we hypothesize are clustering by the semantics of the input prompt-query pair–head (27,5). To perform the dimensionality reduction we vertically stack the activations of a particular head across token positions and project them onto 2D with t-SNE (van der Maaten & Hinton, 2008) coloring by task.

Score-per-token Heatmap. We display the un-aggregated $\rho_{token}(i)$ values for each token re-arranged to convey spatial positioning (equivalent to the 2x2 prompting grid). That is, for the heads of interest, we report the $\rho_{token}(i)$ values that the particular individual head encapsulates–displayed on a heatmap. We place the CLS token as the lone square on the top left corner; following this we have the values corresponding to the four different quadrants (x_s, y_s, x_q, y_q). It is important to note that in the encoder there are no tokens that correspond to the y_q (bottom right) quadrant as these are incorporated as mask tokens after the encoder; hence, for visualization purposes we display their value to be marked as X. We observe that within a single head different tokens can take a wide range of values. Interestingly, there appears to be a consistency in values across the tokens in particular quadrants which motivated the decision to group token patching by quadrants.

Quantitative Clustering Analysis. To further support the observation of high and low quality clustering based off of the scoring value, we report the Silhouette Score and Davies-Bouldin score for the four highest ranked heads, 5 randomly sampled heads, and the three lowest ranked heads (see Table 2). We observe that attention heads scored highly by our methodology display high quality clustering scores from the Silhouette Score and Devies-Bouldin Score and vice-versa for those scored lowly. Finally, the randomly selected attention heads received intermediate scores.

7.3. Implementation Details

MAE-VQGAN (Bar et al., 2022). An MAE (He et al., 2021) with a ViT-L (Dosovitskiy et al., 2021) backbone, which has 24 encoder blocks and 8 decoder blocks, with 16 attention heads in every layer and a patch size of $16x16$ and input resolution of $224x224$. The decoder predicts a distribution over a VQGAN (Esser et al., 2021) codebook to output images with better visual quality. We used the pretrained checkpoint from (Bar et al., 2022) trained over the Computer Vision Figures (Bar et al., 2022) dataset and ImageNet (Russakovsky et al., 2015).

One-shot Prompting. We follow the basic one-shot setup in (Bar et al., 2022). Specifically, we construct a grid-like image structure with an input-output demonstration, a query, and a masked output region which are embedded into a $2x2$ grid. We feed this grid image to the model to obtain the output prediction which we use for evaluation purposes.

Zero-shot Prompting. Similarly to one-shot with the key difference that only the query is embedded into the grid image, in the same bottom left quadrant position as in the one-shot setting. The model is then used to reconstruct only the part that corresponds to the output. Specifically, we patchify the query image at a resolution of 112x112 and apply to it the positional encodings of the bottom left quadrant. The patches are then fed into the encoder and are processed by the decoder together with the mask tokens that correspond to the bottom right quadrant to obtain the result. Note that in the zero-shot setting we also patch intermediate activations by task vectors.

REINFORCE (Ours). We utilize the REINFORCE (Williams, 1992) algorithm of the policy gradients model family in a reinforcement learning environment to select where to patch the task vectors. We model each patch position as a Bernoulli random variable initialized to the sigmoid function value at -1.0, where each patch position is a grouping of token positions i for each individual attention head into 3 groups: CLS token, bottom left quadrant, and bottom right quadrant. Upon each iteration of the REINFORCE algorithm, we sample 32 times from the Bernoulli distribution for each image (fixed at 10 for all experiments) and perform a patching procedure into the positions sampled with a value equal to 1. This results in a total of 320 executions of zero-shot MAE-VQGAN per iteration. Then, we optimize the Bernoulli parameters as outlined in 2.2 with Adam (Kingma & Ba, 2017) using a learning rate of 0.1. We execute the algorithm for a total of 600 steps and select the checkpoint at intervals of 50 steps with the best evaluation on a held out test set.

Greedy Random Search. We compare our methodology to an iterative greedy random search algorithm (GRS) used to select task vectors based on the activation scoring metric proposed in Section 2.1. This serves as a baseline and is outlined in the Supplementary Materials (Section 8.1).

Causal Mediation Analysis. We compare our methodology with the Causal Mediation Analysis methodology as presented by Todd et al. (2023) as a baseline. We select the top 25% of activations with the highest causal score across 10 images.

7.4. Downstream Tasks

Foreground Segmentation. For model evaluations, we use the segmentation masks included in Pascal-5i (Shaban et al., 2017), and report the mean IOU (mIOU) metric.

Low Light Enhancement. To obtain paired input-output data, given a Pascal-5i (Shaban et al., 2017) image, we multiply the color channels by a factor of 0.5 and define the result as the task input and the original un-scaled image as the output. We report the Mean Squared Error (MSE) metric of the prediction with the label.

Inpainting. To obtain input-output pairs given an image, we randomly mask a square region in height/width equal to 25% of the original image (resulting in a black square of 1/8 the area) and define this as the input, while using the original image as the output. For evaluation, we report the MSE metric.

Colorization. To obtain input-output pairs given an image, we convert the original image to grayscale and denote it as the input, and have the output be the original colorful image. To evaluate performance we report the MSE metric.

7.5. Ablations

In this section we describe the set of experiments conducted to ascertain the particular implementation details of our REINFORCE (Williams, 1992) method, validating our design choices.

Table 3. **Isolating Task Locations.** Patching into Encoder only, Decoder only, and Both

	Segmentation ↑			
Model	Split 0	Split 1	Split 2	Split 3
Ours Both (Task-specific)	**0.35**	**0.35**	**0.31**	**0.29**
Ours Encoder (Task-specific)	0.09	0.14	0.14	0.13
Ours Decoder (Task-specific)	0.32	0.34	0.29	0.29

Task Vectors Location in Encoder vs. Decoder. We hypothesize that task implementation occurs in a distributed fashion across multiple parts of the network, necessitating interventions in both the encoder and decoder to induce task implementation in the zero-shot scenario. Hence, we test this by executing our method by restricting interventions to the encoder only, the decoder only, and allowing for interventions throughout the whole network. We report the mIoU on the four splits for the Segmentation task, seeking to find if interventions in both parts of the model are required for appropriate task implementation.

We report our results on isolating the set of possible interventions to the encoder only, decoder only, in contrast to allowing interventions throughout the whole network. We can observe that in-context task learning builds upon both model components. The decoder, however, is more important. It is clear that intervening in both components is crucial for task implementation as we hypothesize that it is computed in a distributed fashion with cascading higher order effects through the network where early interventions have strong downstream effects (see Table 3).

Patching Granularity. We explore different intervention granularities by mapping each token position to a group thus decreasing the dimensionality at which we perform interventions. With this in mind, we can further reduce the size of the search space of the optimization by grouping potential patching positions $i = (l, m, k)$ into spatial quadrants or individual attention heads. We seek to find the optimal granularity at which we maintain precise interventions but manage to reduce the search space nonetheless. We execute our method with three granularity levels (individual tokens, quadrants, attention heads) and report the performance on the four tasks.

Motivated by the emergence of quadrants in the per-token scoring visualization, we explore the optimal granularity at which to group the tokens to reduce the dimensionality of the search space. In the tasks of Segmentation and Colorization, grouping by quadrants results in better performance; whereas in the tasks of Lowlight Enhancement and In-painting, maintaining full patching granularity at the token level results in better performance (see Table 5).

7.6. Visualizations

We provide a wider selection of examples comparing our zero-shot task vector patching methodology to baselines. Alongside each figure, we accompany it with an according analysis.

First, we visualize the task-specific models of our methodology alongside the original MAE-VQGAN, and the CMA and GRS baselines (see Figure 4). Secondly, we visualize the task-specific and multitask variants of our methodology in comparison to CMA (see Figure 5).

Qualitative Analysis for Segmentation. In Figure 6, we compare our methodology and GRS task specific and multitask methods to the original one-shot MAE-VQGAN performance on the task of Segmentation. It appears that our method is good at segmenting the coarse and fine details of the object of focus. In many cases, the segmentations generated by the original MAE-VQGAN suffer from holes or incomplete masks. In contrast, our method outputs consistent and coherent masks. On the otherhand, the GRS method suffers with particular details especially observable when attempting to segment an animal's ear or leg. However, in many such cases it performs better than MAE-VQGAN at getting the general shape of objects.

Qualitative Analysis for Lowlight Enhancement. In Figure 7, we compare our methodology and GRS task specific and multitask methods to the original one-shot MAE-VQGAN performance on the task of Lowlight Enhancement. It appears that the GRS method suffers in maintaining the visual qualities of the query image. However there are many cases where MAE-VQGAN assigns bright colors which is likely due to the particular prompt in use and the inherent ambiguities of the task. On the otherhand, our method–particularly the multitask variant–ouputs consistently better results with accurate visual qualities. In some cases our method produces somewhat muted or blurry results which may be a consequence of using MSE

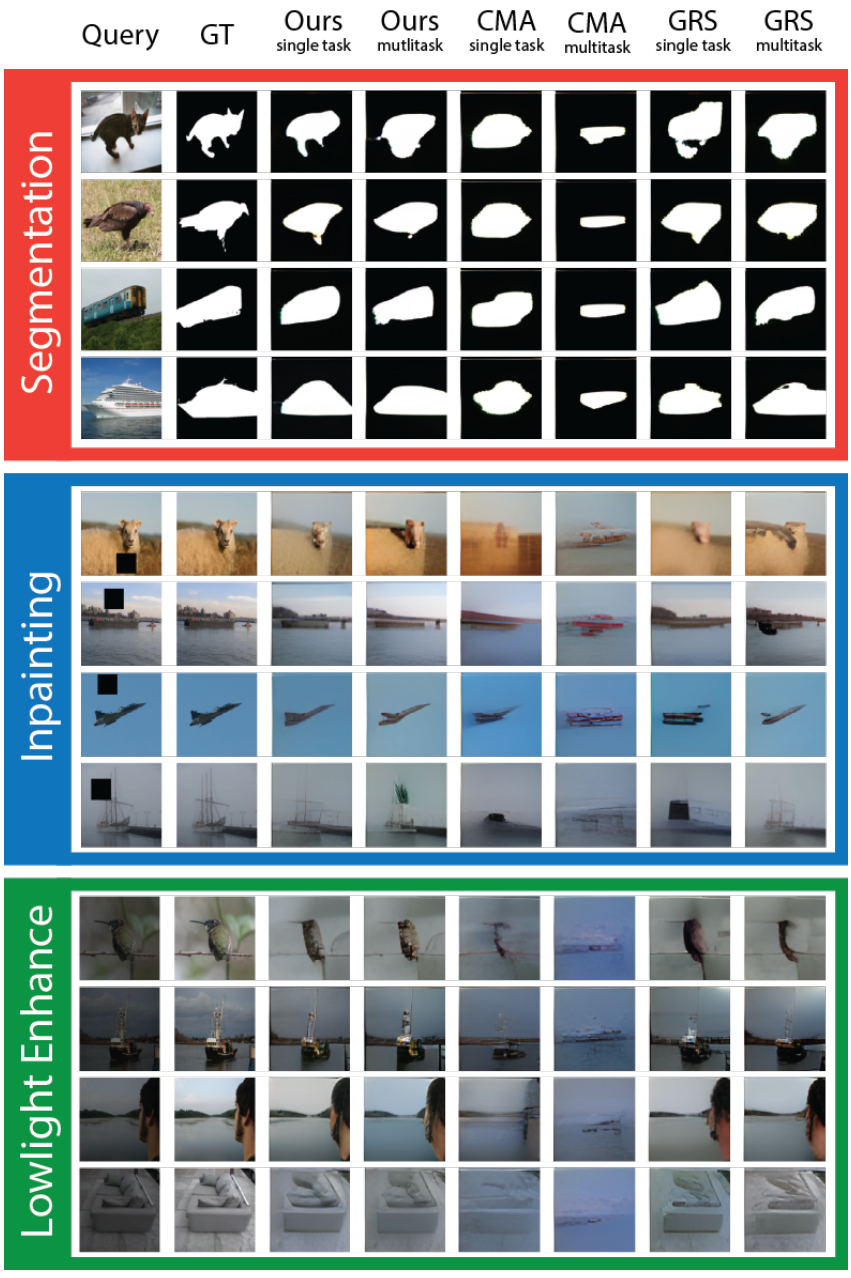

Figure 4. **Qualitative Examples.** We qualitatively compare the task-specific and multitask variants of our methodology with the CMA and GRS baselines. Our patching methodology performs better than the original MAE-VQGAN model.

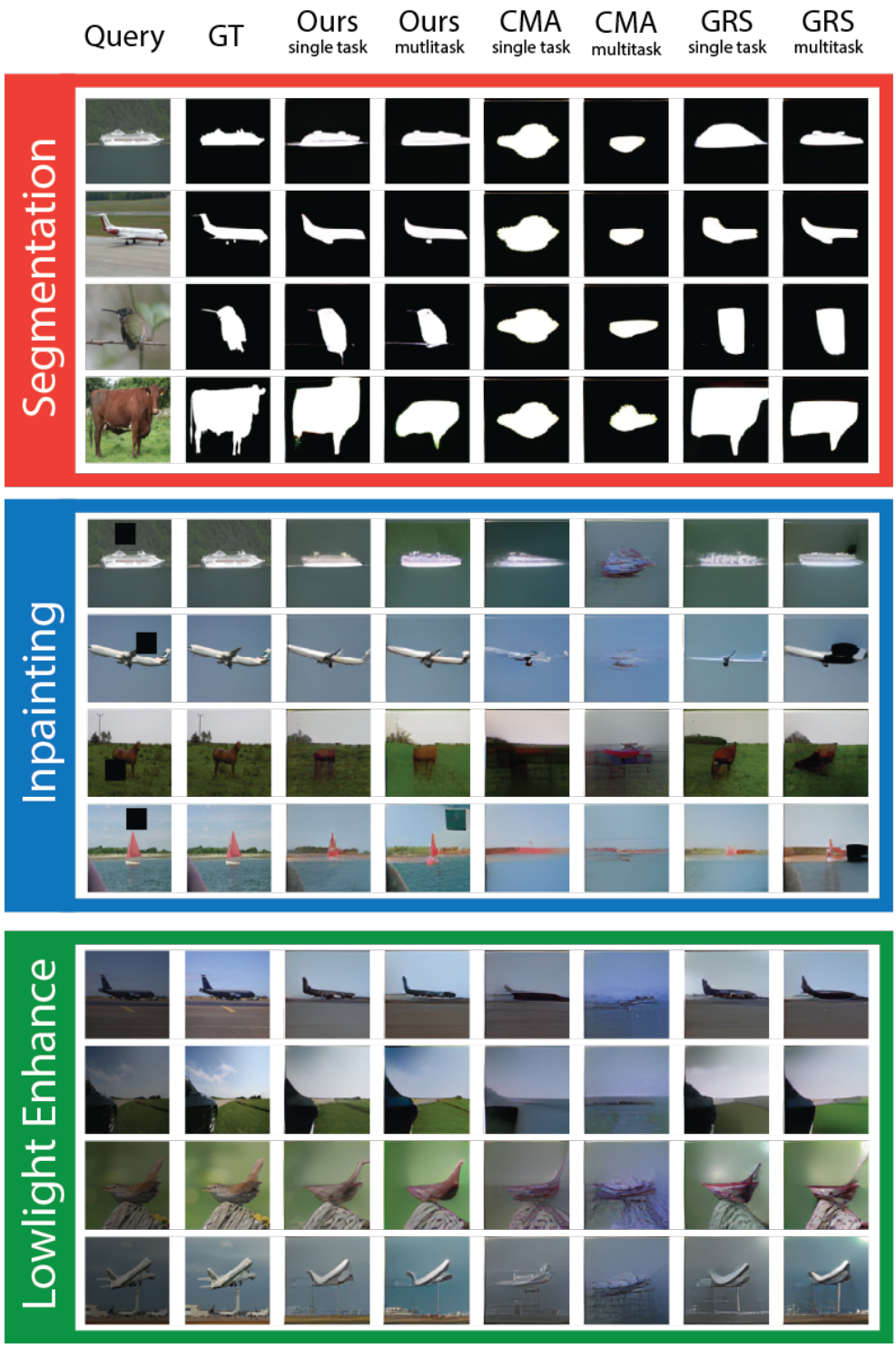

Figure 5. **Qualitative Examples.** We qualitatively compare the task-specific and multitask variants of our methodology with the CMA and GRS baselines.

Table 4. **Quantitative Analysis.** Results comparison across different tasks and splits, indicating the effectiveness of our task specific model.

	Segmentation ↑				Lowlight Enhancement ↓			
Model	Split 0	Split 1	Split 2	Split 3	Split 0	Split 1	Split 2	Split 3
Original MAE-VQGAN	0.35	0.38	0.33	0.29	0.70	0.66	0.73	0.65
Random Quadrants	0.08	0.25	0.16	0.19	4.30	2.40	3.50	1.80
Random K Layers	0.09	0.10	0.09	0.08	1.80	1.80	1.90	1.80
Top Quadrants	0.11	0.17	0.16	0.16	4.50	5.00	5.10	4.90
CMA (Task-specific)	0.23	0.25	0.22	0.22	0.76	0.83	0.88	0.83
CMA (Multitask)	0.18	0.14	0.14	0.14	1.2	1.5	1.5	1.4
GRS (Task-specific)	0.33	0.35	0.30	0.30	0.56	0.61	0.63	0.60
GRS (Multitask)	0.33	0.35	0.31	0.30	0.47	0.52	0.56	0.51
Ours (Multitask)	0.35	0.35	0.31	0.29	0.46	0.49	0.53	0.49
Ours (Task-specific)	**0.38**	**0.38**	**0.33**	**0.32**	**0.41**	**0.46**	**0.50**	**0.46**
	Colorization ↓				In-painting ↓			
Model	Split 0	Split 1	Split 2	Split 3	Split 0	Split 1	Split 2	Split 3
Original MAE-VQGAN	0.59	0.62	0.66	0.60	0.49	0.55	0.61	0.55
Random Quadrants	2.10	4.10	4.30	1.60	3.80	1.20	1.90	2.50
Random K Layers	0.54	0.57	0.60	0.56	0.72	0.89	1.00	0.89
Top Quadrants	3.80	4.40	4.30	4.50	3.70	3.90	4.10	3.90
CMA (Task-specific)	0.79	0.92	0.96	0.91	1.6	1.8	1.9	1.7
CMA (Multitask)	1.02	1.2	1.2	1.1	1.1	1.3	1.3	1.2
GRS (Task-specific)	0.52	0.56	0.59	0.55	0.52	0.56	0.65	0.59
GRS (Multitask)	0.53	0.57	0.61	0.56	0.55	0.61	0.65	0.61
Ours (Multitask)	0.45	0.51	0.55	0.50	0.53	0.56	0.59	0.55
Ours (Task-specific)	**0.40**	**0.46**	**0.50**	**0.45**	**0.45**	**0.49**	**0.51**	**0.47**

Table 5. **Optimal Patching Granularity.** Patching into Tokens (T), Quadrants (Q), or Heads (H)

	Segmentation ↑				Lowlight Enhancement ↓			
Model	Split 0	Split 1	Split 2	Split 3	Split 0	Split 1	Split 2	Split 3
Ours T (Task-specific)	**0.38**	**0.37**	**0.33**	**0.32**	0.44	0.50	0.54	0.50
Ours Q (Task-specific)	0.36	0.36	0.32	0.31	**0.41**	**0.46**	**0.50**	**0.46**
Ours H (Task-specific)	0.24	0.27	0.23	0.24	0.83	0.97	1.02	0.95
Ours T (Multitask)	0.34	0.34	0.30	0.29	0.47	0.51	0.55	0.51
Ours Q (Multitask)	**0.35**	**0.35**	**0.31**	**0.29**	**0.46**	**0.49**	**0.53**	**0.49**
Ours H (Multitask)	0.26	0.27	0.24	0.24	1.01	1.12	1.19	1.10
	Colorization ↓				In-painting ↓			
Model	Split 0	Split 1	Split 2	Split 3	Split 0	Split 1	Split 2	Split 3
Ours T (Task-specific)	**0.40**	**0.46**	**0.50**	**0.45**	0.46	0.49	0.51	0.48
Ours Q (Task-specific)	0.41	0.47	0.51	0.47	**0.45**	**0.49**	**0.51**	**0.47**
Ours H (Task-specific)	0.75	0.88	0.94	0.86	0.78	0.92	0.96	0.88
Ours T (Multitask)	0.45	0.51	0.56	0.52	0.53	0.57	0.60	0.56
Ours Q (Multitask)	**0.45**	**0.51**	**0.55**	**0.50**	**0.53**	**0.56**	**0.59**	**0.55**
Ours H (Multitask)	0.89	1.02	1.08	1.0	0.85	1.02	1.07	0.98

in the pixel space as supervision, but nonetheless reports better quantitative performance.

Qualitative Analysis for In-painting. In Figure 8, we compare our methodology and GRS task specific and multitask methods to the original one-shot MAE-VQGAN performance on the task of In-painting. We observe that our method method consistently outperforms the original model. However, it appears that the GRS task-vector patching method–once again–suffers in maintaining the higher frequency components of the query image; it appears to reduce the contrast of the image and reduce saturation. However, there are many such cases where the original MAE-VQGAN one-shot technique

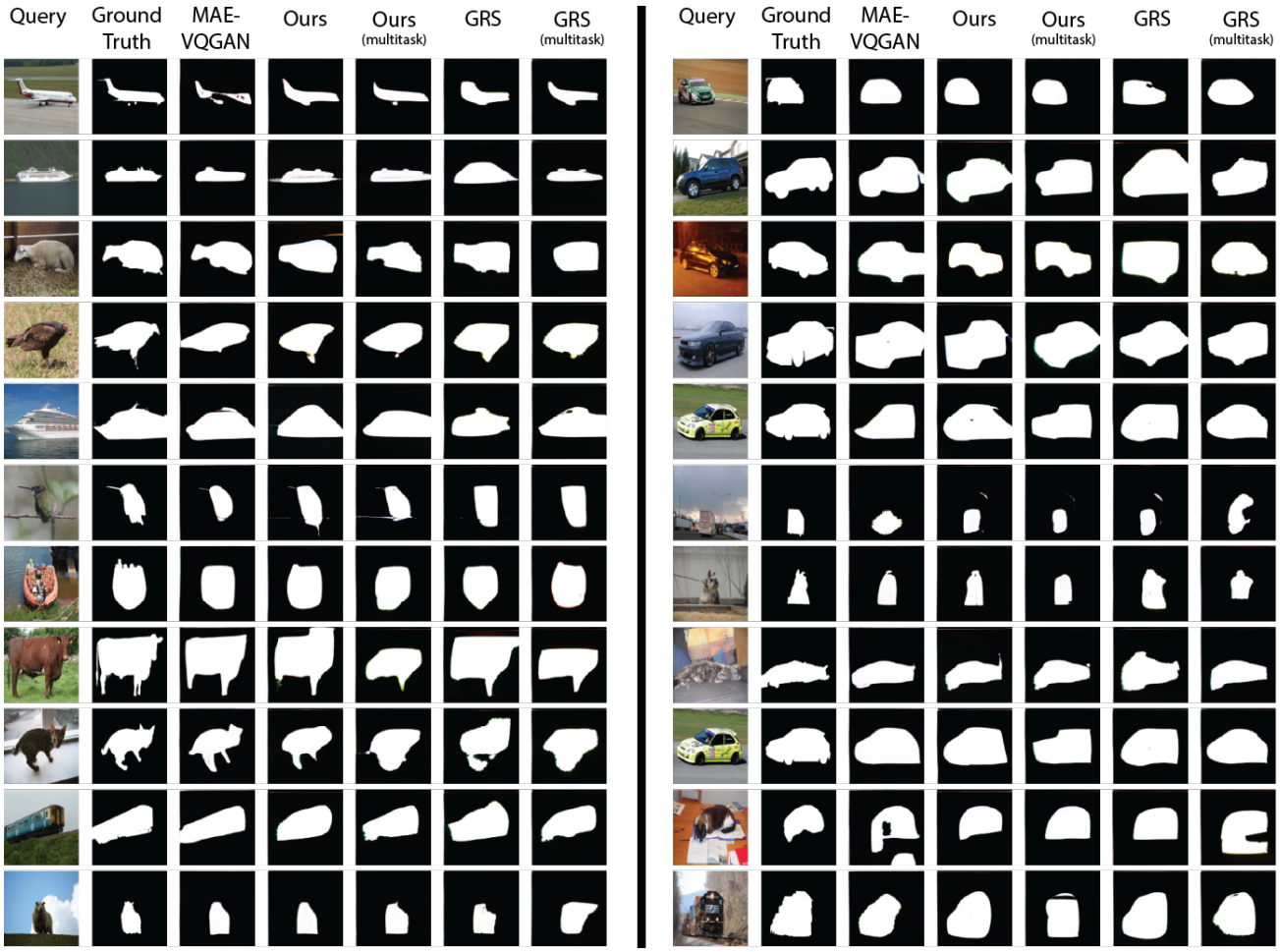

Figure 6. **Our Results on Segmentation Task**

fails to appropriately implement the task while our method succeeds. The original model's performance depends heavily on the specific prompt used which may be the root cause of failures while task-vector patching succeeds.

Qualitative Analysis of Ablations. Finally, we present the qualitative analysis of the different ablations for the Segmentation task in Figure 9. The benefits of observing the visual features of the different ablations in addition to quantitative analysis becomes clear when comparing the Decoder Only and Encoder Only columns. Here it is clear that patching into decoder is of upmost importance in relation to patching into the encoder; the distinction is clear when observing qualitative features. In the end, it is both parts in synchrony which allow for the implementation of in-context learning.

8. Greedy Random Search baseline

8.1. Selecting Task Vectors via Greedy Random Search

For every task j we apply the following algorithm to obtain a task specific model.

Input. The mean activations $\{\mu_{i,j}\}$, scoring function $\rho_{layer}(\cdot)$, pretrained visual prompting model $F(\cdot)$, and an evaluation set.

Initialization. Initialize a set of binary indicators $\{\alpha_{i,j}\}$ for all i, where $\alpha_{i,j} \in \{0, 1\}$ signifies whether the mean task activation $\mu_{i,j}$ is a task vector. Next we describe the algorithm to choose the values of $\alpha_{i,j}$.

For every activation $i = (l, m, k)$ in the top k scoring layers w.r.t $\rho_{layer}(l)$, randomly choose the value of α_i by sampling

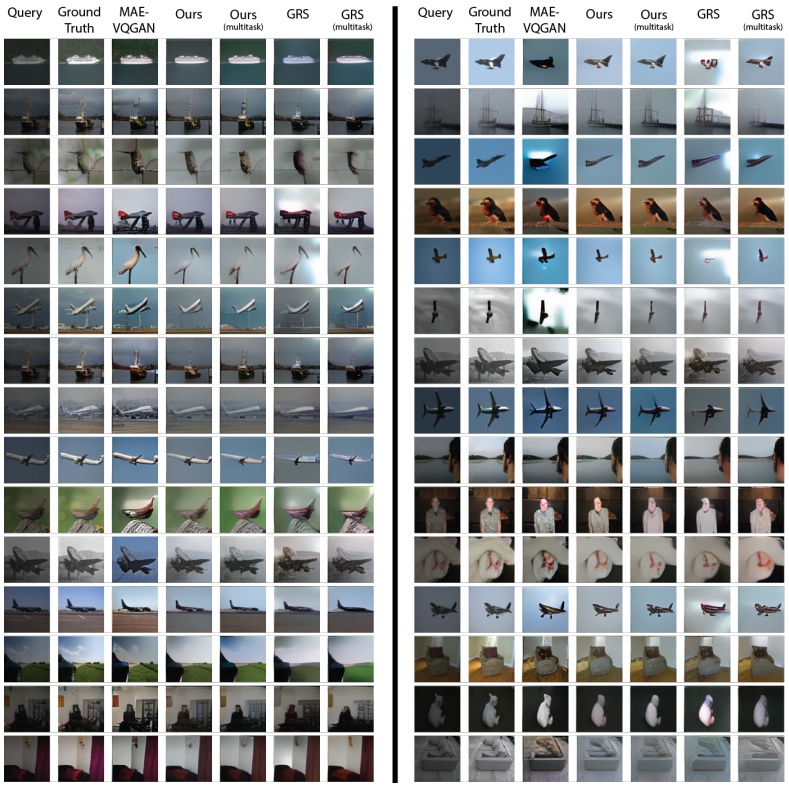

Figure 7. **Our Results on Lowlight Enhancement Task**

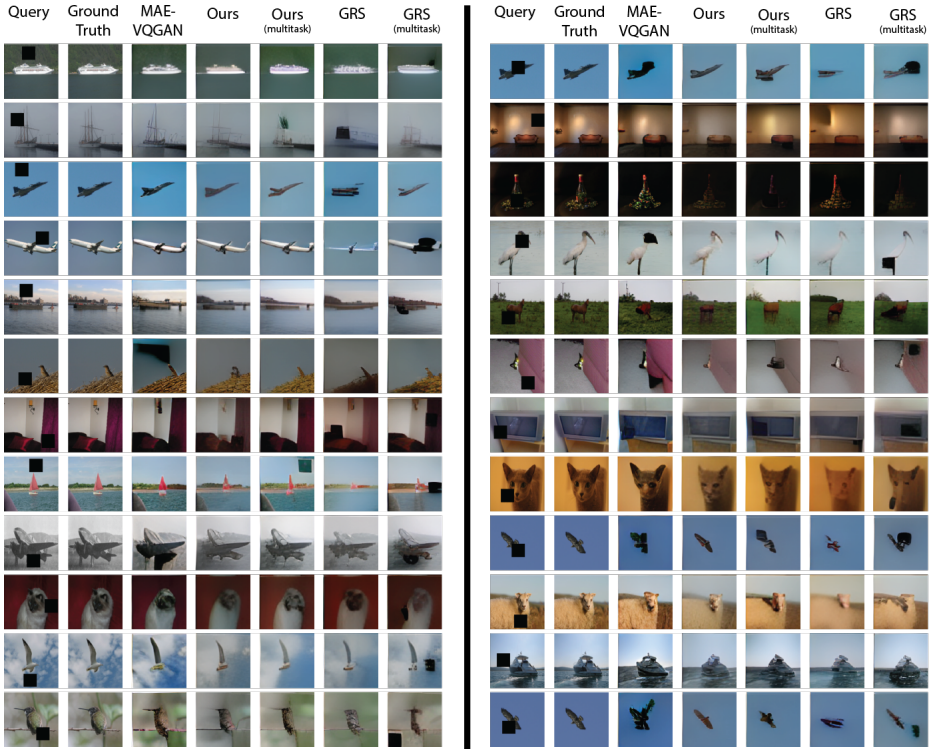

Figure 8. **Our Results on In-painting Task**

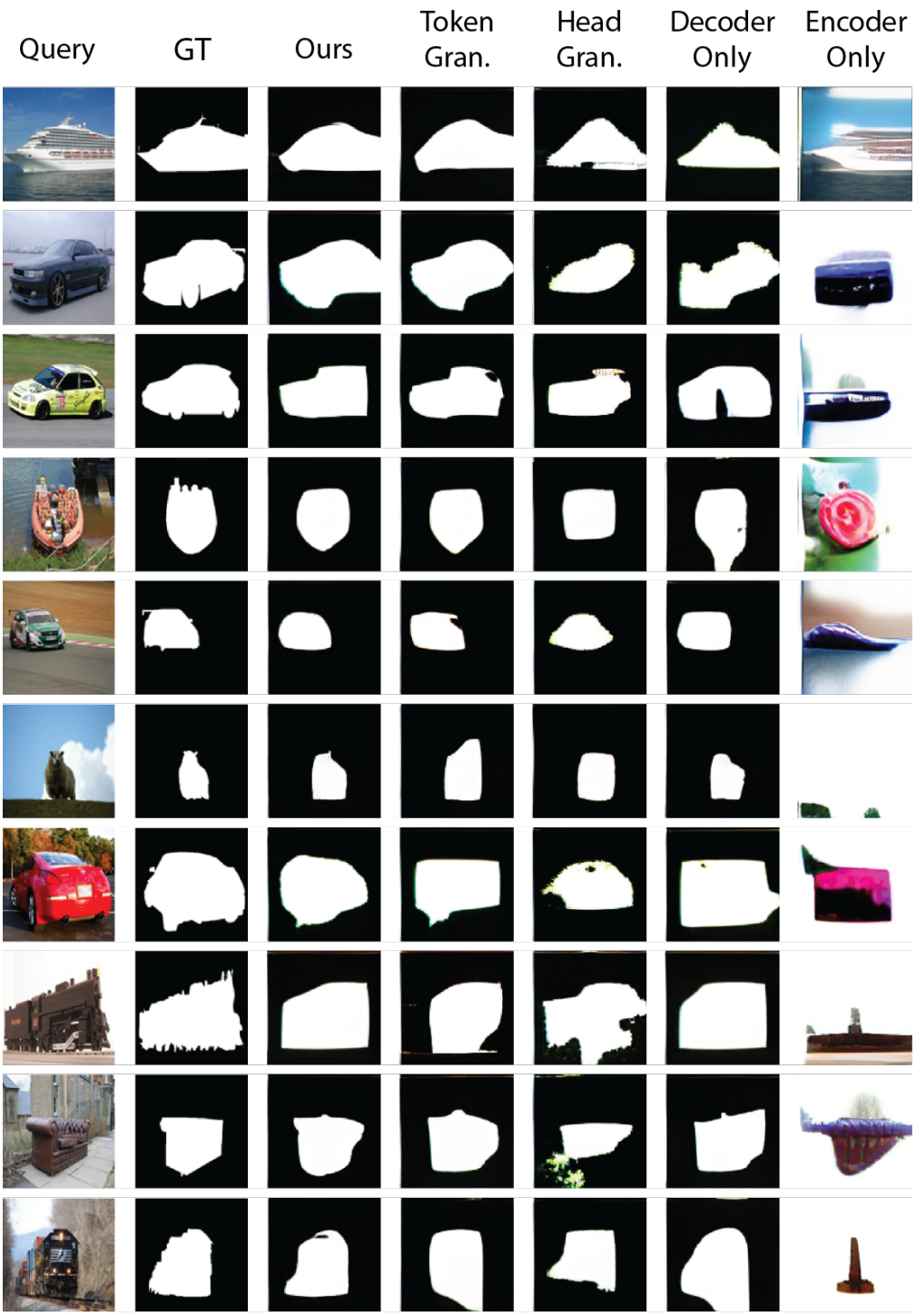

Figure 9. **REINFORCE Ablations for Segmentation Task**

from a Bernoulli random variable with parameter value p. Set the activation vectors to be $z_j = \{\mu_{i,j} | \forall ij \text{ if } \alpha_{i,j} = 1\}$. Evaluate the now task specific model $F(\cdot|z_j)$ on a held out validation set. Run for T trials and for every i, j set the values of $\alpha_{i,j}$ to be the values from the most successful trial.

Greedy Search. Iterate over l in the top k scoring layers sorted by $\rho_{layer}(l)$ from high to low, pick activation $i = (l, m, k)$, flip the value $\alpha_{i,j}$ and evaluate the validation score for $F(\cdot|z_j)$. After having evaluated each flip of the $\alpha_{i,j}$ on the particular layer l we keep the $\alpha_{i,j}$ which performed best or continue to the next layer if the performance did not improve.

Termination. When one search loop across the k layers results in no changes - or after $10k$ iterations, the search has thus converged and we return the single task model $F(\cdot|z_j)$.

This procedure is outlined for every token, attention head, and layer. However, it is possible to apply it in different levels of granularity. For example, by patching group of tokens from the same quadrant, patching all the tokens in an attention head, or patching the entire layer. We discuss these design choices in the next section.

8.2. Greedy Random Search Implementation Experiments

In this section we describe the set of experiments conducted to ascertain the particular implementation details of the greedy random search, validating the design choices.

Implementation Details We search through the top $k = 17$ layers ranked by Activation Scoring. During the initialization phase we sample $\alpha_{i,j} \in \{0, 1\}$ from a Bernoulli distribution with a parameter of $p = 0.3$ (probability of selecting 1) and evaluate performance. We repeat this for $T = 100$ trials and continue with the best performing $\alpha_{i,j}$. Furthermore, we perform a grouping of token positions i in each individual attention head into 3 groups: CLS token, bottom left quadrant, and bottom right quadrant. This serves to further reduce the search space. We use a set of 10 training images to supervise the search. These design decisions are further validated through ablation experiments.

Selecting Initialization Parameters. For the initialization of the Greedy Random Search there are two parameters, k which dictates how many layers to search across and the Bernoulli random variable parameter p which dictates the probability at which we set $\alpha_{i,j}$ to be 1 during the initialization phase. The question is, which k value is best at narrowing down the search space without restricting our ability to induce task implementation, and what is the best according p value? We ascertain this by searching for the optimal configuration to initialize the Greedy Random Search. We perform a grid search for k values from 14 to 20, and p values from 0.1 to 0.6 and report the evaluation metric for the Segmentation task on the batch of 10 images. Our goal is to find the (k, p) pair with highest performing random initialization.

Task Vectors Location in Encoder vs. Decoder. Similar to the ablation conducted for our main methodology (via REINFORCE (Williams, 1992)) implementation, we execute the Greedy Random Search for the Segmentation task by restricting interventions to the encoder only, the decoder only, and allowing for interventions throughout the whole network. It is key to note that in order to restrict interventions to the decoder only, which has 8 layers, the k value must be set to 8, whereas for isolating the encoder we can keep the original $k = 17$ value. We report the mIoU on the four splits for the Segmentation task seeking to find if interventions in both parts of the model are required for appropriate task implementation.

Patching Granularity. We execute our Greedy Random Search with three granularity levels, grouping by Quadrants, grouping by Heads, and grouping by Layers, and report the mIoU performance on the four splits for the Segmentation task.

8.3. GRS Implementation Experiments Results

Selecting K. We explore the optimal parameters for a random initialization. We find the best setup to be to constrain the search across the top $k = 17$ layers, sampling quadrants to patch with a probability of $p = 0.3$ (see Figure 10).

Task Vectors Location in Encoder vs. Decoder. We report the results on isolating the set of possible interventions to the encoder only, decoder only, in contrast to allowing interventions throughout the whole network. We can observe that in-context task learning builds upon both model components. The decoder, however, is more important. It is clear that intervening in both components is crucial for task implementation as we hypothesize that it is computed in a distributed fashion with cascading higher order effects through the network where early interventions have strong downstream effects (see Table 6).

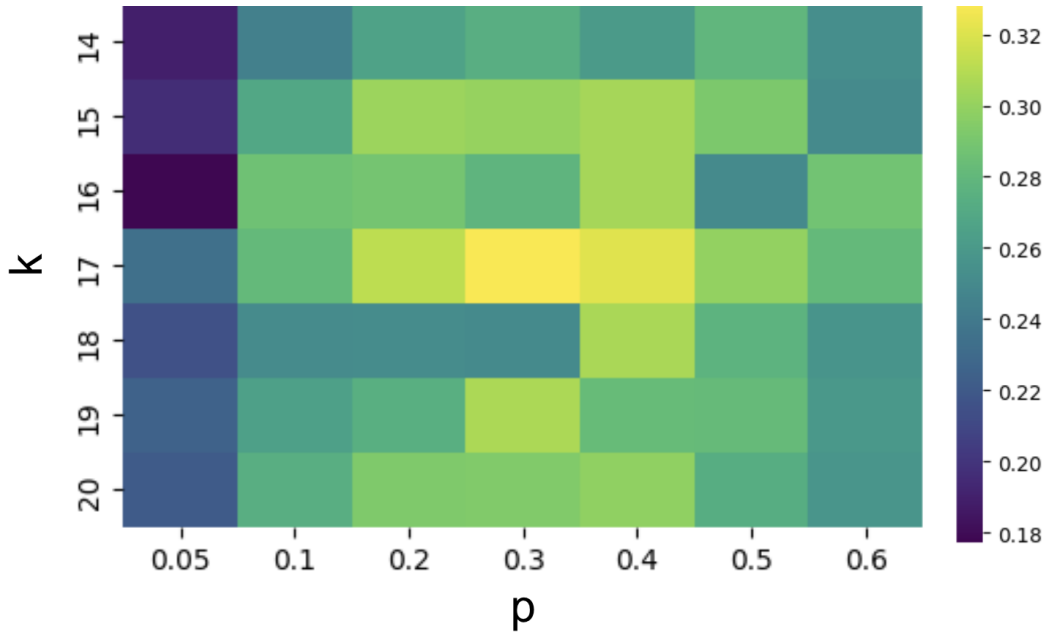

Figure 10. **Selecting Initialization Parameters.** We evaluate Foreground Segmentation mIoU on Pascal 5i using 10 images for different random initialization parameterized by K and p.

Table 6. **Isolating Task Locations.** Patching into Encoder only, Decoder only, and Both

	Segmentation ↑			
Model	Split 0	Split 1	Split 2	Split 3
GRS Both (Task-specific)	**0.33**	**0.35**	**0.30**	**0.30**
GRS Encoder (Task-specific)	0.13	0.22	0.20	0.20
GRS Decoder (Task-specific)	0.26	0.28	0.25	0.25

Table 7. **Optimal Patching Granularity.** Patching into Full Layers, Full Heads, or Quadrants only

	Segmentation ↑			
Model	Split 0	Split 1	Split 2	Split 3
GRS Quadrants (Task-specific)	**0.33**	**0.35**	**0.30**	**0.30**
GRS Heads (Task-specific)	0.15	0.15	0.14	0.13
GRS Layers (Task-specific)	0.28	0.31	0.26	0.27

Patching Granularity. We explore the optimal granularity at which to group the tokens to reduce the dimensionality of the search space. Motivated by the emergence of quadrants in the per-token scoring visualization, and validated by attempting to group by whole attention heads (patching into all the tokens in the attention head) and group by whole layers (patching into all the attention heads of a layer), it is clear that quadrants provide the best trade off between reducing the dimensionality of the search space and performance (see Table 7). It is interesting to note that patching into full layers reduces the search space to a size of 2^{32} whereas attention heads is 2^{512} and quadrants is 2^{768} for the encoder and 2^{384} in the decoder (disregarding top-k layer selection).

8.4. GRS Baseline Qualitative Comparisons

We provide a wider selection of examples comparing our GRS zero-shot task vector patching methodology in comparison to **1) a selection of baselines, and 2) our ablations.** Alongside each figure, we accompany it with an according analysis.

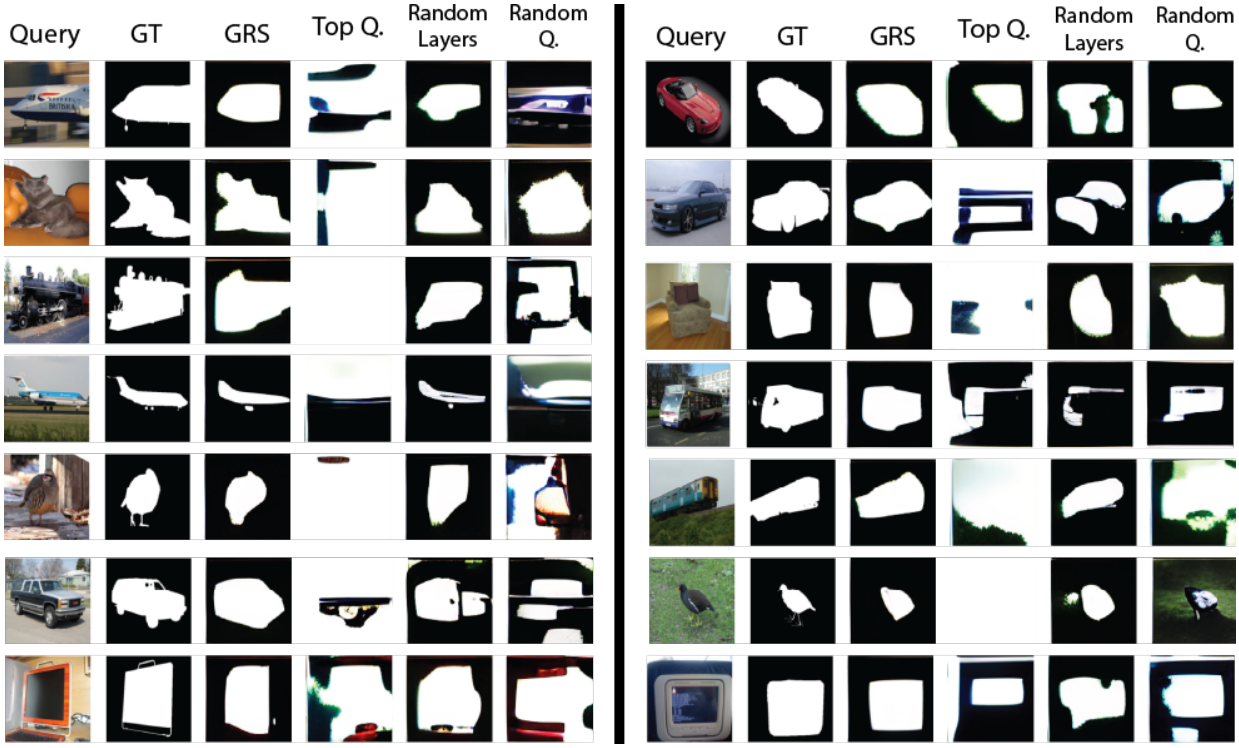

Figure 11. **GRS Baseline Comparison for Segmentation**

In the Figures 11, 12, and 13 we compare our GRS task specific method with a handful of baselines defined in section 3.2. We have abbreviated Top Quadrants as Top Q, Random K Layers as Random Layers, and Random Quadrants as Random Q. It is clear that Top Quadrants struggles to output coherent completions. We believe this to be because of the need of patching into positions of different purposes other than task implementation such as positions that encode the input-output structure that a one-shot prompt provides. Further opportunities for exploration could include other scoring terms that take into account structural information provided by different prompt orientations. Random K Layers performs surprisingly well due to the efficiency of the Greedy Random Search but nonetheless does not reach the performance of using our scoring mechanism to select the top K layers. Finally Random Quadrants struggles to complete coherent results.

Finally, we present the qualitative analysis of the different ablations for the Segmentation task in Figure 14. Similarly to our main method (via REINFORCE (Williams, 1992)), it is clear that patching into decoder is more important than patching into the encoder but in the end, it is both parts in synchrony which report the best performance. Furthermore, it is clear that the optimal granularity for patching is at a quadrant level. We find it counter intuitive that layer-level patching performs better than head-level patching–as one would assume that a finer granularity provides better accuracies. However, we believe that by grouping per layer we significantly reduce the search space (by a factor of 16) which reduces the probability of falling into a local optimum; whereas grouping by head, we suffer from a reduced precision but do not gain the benefits of a

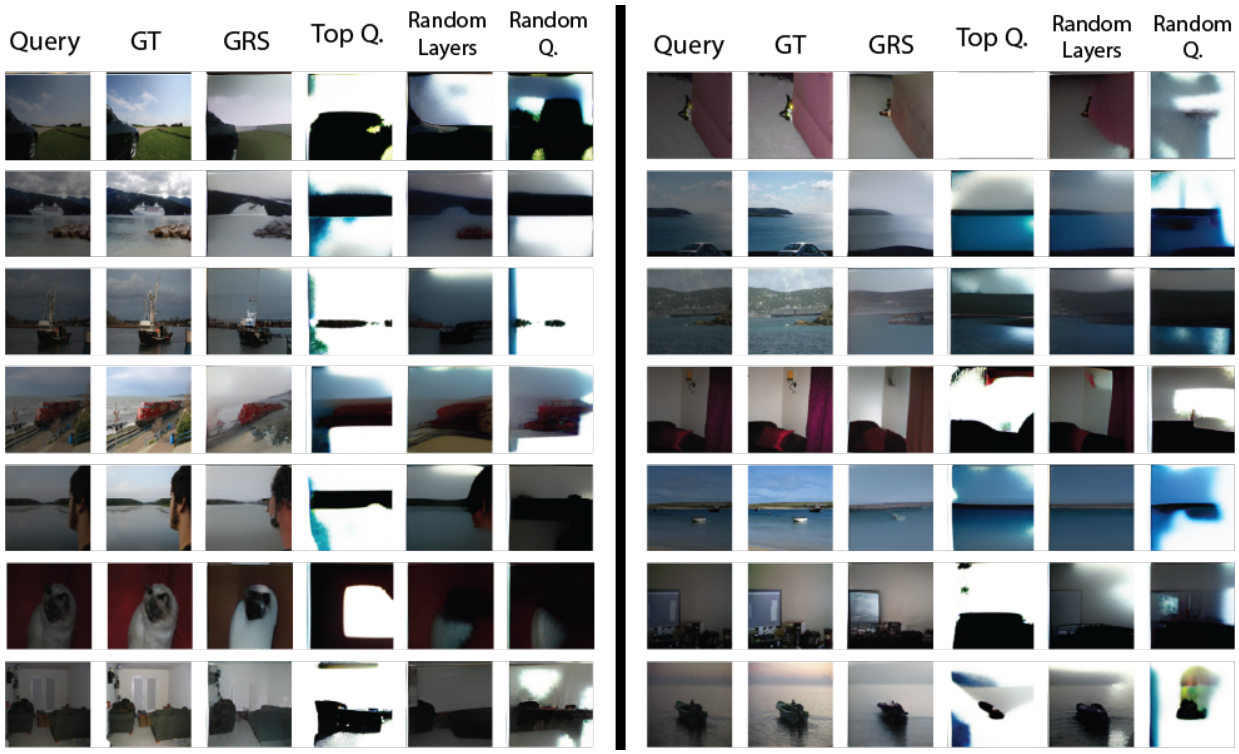

Figure 12. **GRS Baseline Comparison for Low light Enhancement**

reduced search space magnitude (factor of 2 for encoder and factor of 3 for decoder when grouping by head instead of 16 when grouping by layer). Further exploration in this direction is of interest.

9. Limitations

Our focus was in identifying Task Vectors, but we hypothesize that there might exist other important vector-types such as image structure activations that capture locations, input-output placements, and left-to-right ordering. During the optimization with REINFORCE (Williams, 1992) we evaluate the performance using the MSE metric for all tasks (except Segmentation which uses mIoU) after decoding the predicted VQGAN tokens. Instead, it may be possible to directly evaluate the model in VQGAN token space using cross entropy loss which may lead to more accurate results. We leave these extension for future works.

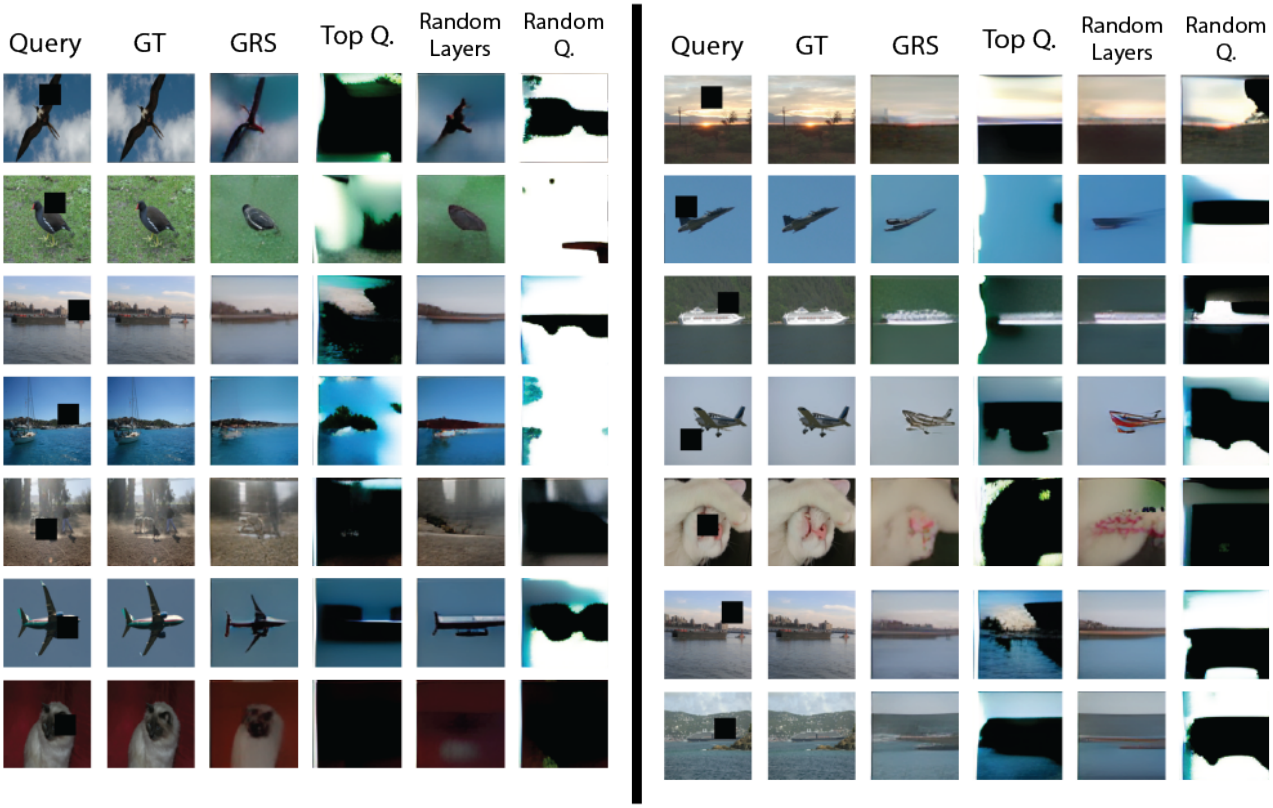

Figure 13. **GRS Baseline Comparison for In-painting**

Figure 14. **GRS Ablation for Segmentation Task**