Tuning Computer Vision Models With Task Rewards

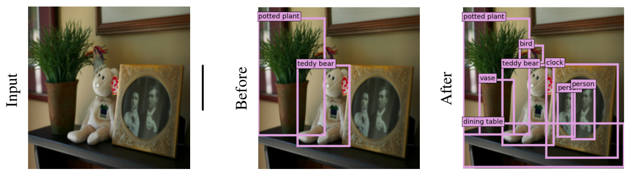
André Susano Pinto [*] [1] **Alexander Kolesnikov** [*] [1]
Yuge Shi [1] [2] **Lucas Beyer** [1] **Xiaohua Zhai** [1]

Abstract

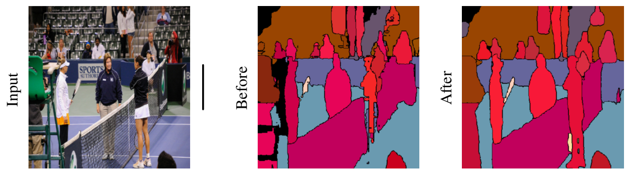
Misalignment between model predictions and intended usage can be detrimental for the deployment of computer vision models. The issue is exacerbated when the task involves complex structured outputs, as it becomes harder to design procedures which address this misalignment. In natural language processing, this is often addressed using reinforcement learning techniques that align models with a task reward. We adopt this approach and show its surprising effectiveness to improve generic models pretrained to imitate example outputs across multiple computer vision tasks, such as object detection, panoptic segmentation, colorization and image captioning. We believe this approach has the potential to be widely useful for better aligning models with a diverse range of computer vision tasks.

1. Introduction

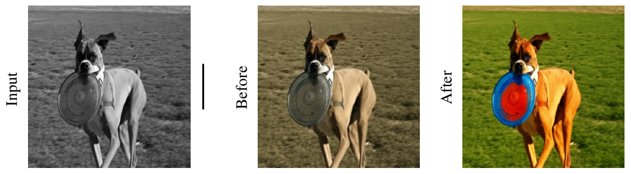
The main criteria for success when dealing with complex outputs in computer vision is not how well the model optimizes the training objective, but whether the predictions are aligned with the task risk, i.e. the model's performance on the intended usage. In order to improve this alignment, as a community we iterate on model architectures, data, optimization, sampling procedures, post-processing, etc. As an example, in the context of object detection, researchers use non-maximum suppression post-processing (Ren et al., 2015; Lin et al., 2017), set-based global loss (Carion et al., 2020) or even alter the input data (Chen et al., 2022) to obtain models with improved behavior at test time. Although these approaches deliver significant gains, they are often highly specialized to the task and method at hand.



(a) Optimize mAP: $39 \rightarrow 54$, results in a much higher recall and learns box prediction confidences.



(b) Optimize PQ: $43.1 \rightarrow 46.1$, removes many incoherent predictions, especially for small-scale objects.



(c) Optimize "colorfulness" score: $0.41 \rightarrow 1.79$, improves color diversity and saturation.

Figure 1. By tuning a strong, pretrained model with a reward that relates to the task, we can significantly improve the model's alignment with the intended usage.

This problem is not new. It has been extensively studied by the natural language processing (NLP) and reinforcement learning (RL) fields, where it is notoriously hard to formulate an optimization objective for tasks with less tangible goals, such as translation (Kreutzer et al., 2018) or summarization (Stiennon et al., 2020). A popular approach when dealing with this type of problem is to *learn to imitate example outputs, followed by reinforcement-learning to align the model with a reward function*. Using this approach, the NLP field is now producing exciting results with systems that use large pretrained language models and rewards defined by human feedback to tackle tasks that were otherwise hard to specify (Ouyang et al., 2022). Additionally, the

[*]Shared first authorship and leadership. [1]Google DeepMind, Zürich, Switzerland [2]Work done during internship while being a PhD student at the University of Oxford. Correspondence to: André Susano Pinto <andresp@google.com>, Alexander Kolesnikov <akolesnikov@google.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

same approach is widely adopted for the image captioning task (Rennie et al., 2017), where CIDEr (Vedantam et al., 2015) is used as a reward. Despite that, to the best of our knowledge, reward optimization has not been previously explored for improving and bridging the gap of generic models on (non-textual) computer vision tasks.

In this work, we demonstrate that tuning a pretrained model with a reward function using REINFORCE (Williams, 1992) works out-of-the-box for a wide range of computer vision tasks. We illustrate some of our key results in Figure 1, highlighting both quantitative and qualitative improvements brought by reward optimization for object detection, panoptic segmentation, and image colorization. The simplicity and effectiveness of our approach on a diverse set of computer vision tasks demonstrates the versatility and adaptability of generic models with no task specific components. Although in this work we mostly use rewards in the form of evaluation metrics, we believe these initial results show promising paths to optimizing computer vision models with more complex and harder to specify rewards, e.g. human feedback or holistic system performance.

2. Related Work

Optimizing Computer Vision Metrics. There is a vast amount of literature in computer vision that sets the goal of optimizing complex non-decomposable or non-differentiable metrics. In this section we highlight some prominent work. Henderson & Ferrari (2017) propose a specialized approach to compute a pseudo-gradient in order to optimize the average precision (AP) metric for object detection. Song et al. (2016) propose a general framework for computing approximate gradients of metrics. In the field of semantic image segmentation, CRF loss (Lafferty et al., 2001) is often used to ensure segmentation mask consistency. However, the gradient of the CRF-based loss is generally intractable to compute, so many approximations (Krähenbühl & Koltun, 2011) or constrained CRF variants (Nowozin et al., 2011; Kolesnikov et al., 2014) were proposed in the literature. In contrast, we propose a generic way to optimize arbitrary rewards that are aligned or coincide with the task risk, for models that are capable of sampling predictions.

Related to our work is Rao et al. (2018), which uses REINFORCE (Williams, 1992) to learn a policy to select detection candidates produced by a Faster R-CNN network. In contrast we show REINFORCE can be used to optimize all aspects of the output for several tasks while using the same modelling decisions and a generic MLE pretraining goal. Another closely related work is Huang et al. (2021), which proposes an online algorithm that approximates the task reward value with a neural network. This differentiable reward approximation is then used to tune a model. In contrast, we suggest to directly optimize the reward function by relying on the well-known log-derivative trick and the ability of the underlying model to sample multiple predictions. Finally, REINFORCE-based tuning has been selectively applied for low-level vision tasks, e.g. for 2D point matching (Bhowmik et al., 2020) or pose estimation (Krull et al., 2017).

Optimizing Text Generation. Ranzato et al. (2015) demonstrate improved results on captioning, translation and summarization tasks by training text models with a mixture of MLE and REINFORCE to optimize the non-differentiable rewards (BLEU and ROUGE). Shen et al. (2015) also optimizes translation for evaluation metrics but by aproximating the posterior distribution with samples. Rennie et al. (2017) show that using independent model samples as baseline and optimizing CIDEr is simple yet highly effective for image captioning. Keneshloo et al. (2019) provide a survey of text tasks and uses of RL in seq2seq models. Recently, more advanced RL techniques incorporating human feedback have been used by Ouyang et al. (2022); Glaese et al. (2022) to align large language models with human intent.

Generalization of Sampled Outputs. Several works such as Ranzato et al. (2015) and Bengio et al. (2015) discuss exposure bias, i.e. the distribution discrepancy of previous tokens between training and generation, as a cause for low sample quality. They explore approaches that include sampling from the model during training. Schmidt (2019) argues that generalization, and not exposure bias, is the underlying issue to address. Stahlberg & Byrne (2019) point out that even when using large beams and exact inference, translation models can fail by considering empty and smaller sentences as more likely. Nucleus sampling (Holtzman et al., 2020) was designed to alleviate sampling degeneration. Leblond et al. (2021) explore different sampling procedures in translation aided by consistency scores (e.g. multilingual BERT). Ramesh et al. (2021) train a model to generate images from text and use a pretrained contrastive model to filter out images inconsistent with the text. Chen et al. (2022) train a generative model for object detection, however good performance of the model is contingent on example augmentation and modified sampling procedures.

Reinforcement Learning in Vision. Many previously proposed vision models also leverage reinforcement learning algorithms for vision tasks. They generally focus on learning a system that sequentially attends to various parts of the image and does iterative refinement of the outputs. A prominent example is (Mathe et al., 2016), which learns a sequence of image "glimpses" that extract visual features from the specific regions and iterative box predictions. See (Le et al., 2021) for the wide overviews of these type of approaches for object detection and other vision tasks. We largely differ from these approaches, as we do not change the underlying model architecture and instead tune a generic model to optimize the task-specific reward.

Algorithm 1 MLE optimization step

```
function batch_loss($\theta$, $\boldsymbol{x}$, $\boldsymbol{y}$):
    # $n$ is the size of a mini-batch.
    return $\frac{1}{n}\sum_{i=1}^{n}\left(\log P(y^i|x^i;\theta)\right)$
end function

function step_mle($\theta$, $\boldsymbol{x}$, $\boldsymbol{y}$, $\alpha$):
    $G_{mle}$ := $\nabla_\theta$ batch_loss($\theta$, $\boldsymbol{x}$, $\boldsymbol{y}$)
    return $\theta + \alpha G_{mle}$
end function
```

Algorithm 2 Reward optimization step

```
function batch_loss($\theta$, $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{r}$):
    return $\frac{1}{n}\sum_{i=1}^{n}\left(r\log P(y^i|x^i;\theta)\right)$
end function

function step_reward($\theta$, $\boldsymbol{x}$, $\alpha$):
    $\boldsymbol{y}_{sample}$ := sample_model_output($\theta$, $\boldsymbol{x}$)
    $\boldsymbol{y}_{baseline}$ := sample_model_output($\theta$, $\boldsymbol{x}$)
    $\boldsymbol{r}$ := $\mathcal{R}(\boldsymbol{x}, \boldsymbol{y}_{sample})$ - $\mathcal{R}(\boldsymbol{x}, \boldsymbol{y}_{baseline})$
    $G_r$ := $\nabla_\theta$ batch_loss($\theta$, $\boldsymbol{x}$, $\boldsymbol{y}_{sample}$, $\boldsymbol{r}$)
    return $\theta + \alpha G_r$
end function
```

3. Tuning Models With Rewards

Without loss of generality, we formulate a computer vision task as learning a function that maps an input x (in our case an image) to an output represented as a sequence of values $y = [y_1, y_1, \dots, y_n]$ (e.g. sequence of text tokens, sequence of bounding boxes, per-pixel outputs). We assume availability of a dataset of N training examples $D = \{(x^i, y^i)\}_{i=1}^{N}$ sampled from the distribution $\mathcal{D}$. When describing algorithms we use bold $\boldsymbol{x}$ or $\boldsymbol{y}$ to describe a mini-batch of items. Our goal is to learn a conditional distribution $P(y|x,\theta)$ parameterized by θ that maximizes a reward function $\mathcal{R}$, which coincides or closely aligns with the task risk. Formally, we want to solve the following optimization problem

$$\max_{\theta} \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{y\sim P(\cdot|x,\theta)} \mathcal{R}(x,y)\right] \tag{1}$$

Our proposed framework for solving the above problem is very simple, consisting of two steps: (1) model pretraining with maximum-likelihood estimation (2) model tuning for the task risk by maximizing a related reward with the REINFORCE algorithm. We first describe these steps algorithmically and later discuss the intuition and motivation behind the proposed approach.

Maximum-Likelihood Pretraining. We first use the maximum likelihood principle to estimate parameters θ and capture the distribution of training data. This can be done with the gradient descent algorithm by maximizing the log-likelihood $\sum_{i=1}^{N}\log P(y^i|x^i,\theta)$ of the training data. The MLE optimization step is described in Algorithm 1 and is currently the most common way of training a model. We refer to the model resulting from this step as the MLE model.

Reward Maximization With REINFORCE. In order to further tune the MLE model to the task risk, we maximize a related reward function. Note that we optimize the reward when using model outputs and not outputs from the training data. We leverage the REINFORCE algorithm (also known as the "log-derivative trick") to estimate the gradient of the expected reward for a given input x:

$$\nabla_\theta \mathop{\mathbb{E}}_{y\sim P}\left[\mathcal{R}(x,y)\right] = \mathop{\mathbb{E}}_{y\sim P}\left[\mathcal{R}(x,y)\nabla_\theta \log P(y|x;\theta)\right].$$

Note that the unbiased estimate of the right-hand side of this equation can be computed as an average of per-example gradients and does not require the reward function to be differentiable. In order to reduce the variance of this gradient estimate, it is common to subtract a baseline value b (independent of the considered example) from the reward function. In practice, we draw two sample outputs for one training input, use one to estimate the gradient and the other to compute the baseline reward b. For clarity, we provide pseudocode in Algorithm 2.

Discussion. The two optimization steps outlined above have complementary strengths and weaknesses. In practice, neither of them in isolation is sufficient to optimize for a task, but when chained together they work very well.

The first step, model training via the conditional maximum-likelihood estimation, is one the most studied and well-understood approaches in machine learning. There are now very powerful and efficient probabilistic models, e.g. the Transformer encoder-decoder model (Vaswani et al., 2017), that can be trained with MLE and can capture very complex data distributions. However, these type of models have a crucial shortcoming. While they can excel at capturing the distribution of training and test data, they are agnostic of the actual task risk and may not perform sufficiently well in their intended usage.

Thus, we leverage the REINFORCE algorithm to further tune the MLE model to optimize an arbitrary reward function related to the task risk. Crucially, it is sufficient to provide only the numerical value of the reward, without any requirements for the reward functions, such as being differentiable, or being able to run it on a computer (e.g. one can use user feedback as reward). Note that using REINFORCE from scratch in the computer vision tasks we explore is most likely unfeasible due to the large output space and reward sparsity. However, by using a pretrained MLE model, we have a good initial sampling strategy and only need a relatively small number of optimization steps to make quick progress in optimizing the reward function.

4. Practical Applications

In this section we show several applications of the described approach to optimize models for vision tasks. In most cases we use an encoder-decoder architecture with a ViT (Dosovitskiy et al., 2021) encoder to process images and an auto-regressive Transformer decoder to model output distributions. We first pretrain the model using maximum-likelihood estimation then tune it with a task reward. For both steps we use a variant of Adafactor (Shazeer & Stern, 2018) introduced by Zhai et al. (2022) as optimizer and sample greedily at inference time to report results.

It is also important to keep in mind that although in this section we treat existing validation metrics as the task risk, in a real scenario those might differ significantly. In such cases, one might require further validation of the model or iterations on the reward design to guarantee improved performance on the intended usage. Overall our goal is to show that reward optimization is a suitable and general approach to improve computer vision model performance.

4.1. Panoptic Segmentation

Panoptic segmentation (Kirillov et al., 2019) task can be seen as an aggregation of both instance and semantic segmentation and requires a model to produce a coherent scene segmentation, by assigning a label and instance id to pixels. The metric commonly used in related literature is *Panoptic Quality* (PQ). PQ is designed to capture the completeness and detail of predictions and measures. It is computed as a within-class average of mean IoU of matched instances (TP), while penalizing for extra predicted instances (FP) and missed ground truth instances (FN):

$$PQ = \underset{k \in K}{\text{mean}} \frac{\sum_{(p,g) \in TP_k} IoU(p,g)}{|TP_k| + \frac{1}{2}|FP_k| + \frac{1}{2}|FN_k|}$$

MLE Pretraining. We use UViM panoptic model (Kolesnikov et al., 2022) as our MLE pretraining baseline. UViM adopts a ViT-L/16 encoder for 512×512 resolution and a 24 layers auto-regressive decoder. The decoder output is a 256 discrete sequence of 4 k possible tokens which can then be decoded by UViM stage I models into a 512×512 panoptic output. The model was trained using MLE on COCO panoptic dataset.

Tuning for PQ. The PQ computation is not decomposable as a sum of per-example rewards. We opt to use a reward which is the sum of matched IoUs and a negative weight $w = 0.3$ to unmatched predicted instances in an example:

$$reward(y, k) = \left[\sum_{(p,g) \in TP_k} IoU(p,g)\right] - w|FP_k|$$

We use REINFORCE rule to tune the MLE model for this reward, with a batch size of 128 for 30k steps with constant learning rate 10^{-6} after a warmup of 4 k steps. We observe that our tuning procedure significantly improves the MLE model (see table 1). Our visual inspection suggests that the tuned model is better at avoiding incoherent predictions, especially for the small-scale objects, see 1 as an example.

Table 1. Panoptic segmentation results on COCO panoptic validation set after reward optimization.

MODEL	PQ (%)
$\text{UViM}_{512\times512} \rightarrow$ *Ours*	$43.1 \rightarrow 46.1$
$\text{UViM}_{1280\times1280}$	45.8

Note that the task here is quite challenging as we are optimizing a model to sample a discrete sequence with little feedback from a complex reward function. The reward (a scalar) of a model output (a 256-length discrete sequence) is computed by using a neural network to decode the sequence into a 512×512 per-pixel panoptic output which is then compared against the ground truth to approximates a PQ-value per example.

4.2. Object Detection

In the object detection task the goal is to predict a tight bounding box for objects (e.g. *chair* or *pen*) present in an input image. The task is notoriously hard due to the complex nature of the output. Many different approaches, with unique pros and cons, have been proposed in the past. One group of techniques (Ren et al., 2015; Lin et al., 2017) predicts a large redundant collection of boxes and then applies specialized post-processing (non-maximal suppression) at test time. Another approach, proposed by Carion et al. (2020), relies on the set-based global loss during training. Finally, Pix2seq (Chen et al., 2022) propose to use a generative model to directly model likelihood of the training data encoded as a sequence of discrete values (discretized box coordinates and semantic class labels).

A shared shortcoming of all these approaches is that they do not offer an explicit way to obtain a model aligned with the task risk and, instead, rely on design choices that implicitly modulate object detection model properties. For example, Faster-RCNN models use a two stage box prediction approach to better balance positive and negative boxes. Similarly, RetinaNet uses the *focal loss* to achieve the same effect. On the other hand, Pix2seq alters the training data by jittering the ground-truth boxes and adding "fake" boxes to trick the prediction model into outputting more object bounding boxes.

Instead, in our experiments, we use detection-specific rewards to optimize a vanilla detection data likelihood model (similar to Pix2seq's base model). Importantly, this bypasses the need of adopting specialized heuristics to optimize for the standard metrics. We represent a set of bounding boxes as a discrete sequence by discretizing the coordinates in 1000 buckets, plus one token for the class label and one token for the per-box prediction confidence. We use the standard ViT-B/16 as image encoder and 6-layer auto-regressive Transformer decoder (with the same configuration as the ViT-B model). Following our approach we pretrain a MLE model and then tune it with rewards for recall and mAP.

MLE Pretraining. Following the standard practice, we pretrain the model on the *Objects365* dataset (Shao et al., 2019) and further finetune on the COCO (Lin et al., 2014) dataset. The model is pretrained on the *Objects365* dataset for 400 k steps using 256 batch size, with a learning rate of 0.001 and 0.00005 weight decay. We linearly warm up the learning rate for the initial 20 k steps, and then decay it to zero using a cosine schedule. We then finetune the model on COCO, using a smaller learning rate 10^{-4} without weight decay, for 10 k steps. Cosine learning rate schedule with 1 k warmup steps is adopted. We used 640×640 resolution for *Objects365* pretraining and 1280×1280 resolution for COCO finetuning. The resulting MLE model achieves 54.4% average recall@100 and 39.2 mAP score on COCO.

Tuning for Recall. *Average recall @ N* is a popular metric for evaluating object detection models and is expected to correlate with usage in retrieval applications. This metric computes the percentage of object instances in the ground truth that are matched (at a certain IoU threshold) to one of the predicted boxes. At most N predictions per image are allowed. Recall for each IoU threshold and semantic class and is computed independently and then averaged. Our per-image recall reward is implemented as the count of matched ground-truth boxes minus the number of "duplicate boxes" (the boxes that have been matched to an already matched ground-truth box) with 0.3 multiplier.

We tune our MLE model to optimize the recall reward for 100 k steps with the constant learning rate of 10^{-6}. Table 2 demonstrates that the resulting model successfully optimizes the average metric, pushing its value from 54.4% to 68.4%. Note that optimizing for recall alone results in mAP results degrading to 16.8 AP points. This is expected, as optimizing for recall only results in many low-quality boxes. We address this issue in the next section.

Tuning for Mean Average Precision. *Mean average precision (mAP)* is a metric based on the area under the precision-recall curve of predicted instances of each class that get matched with a given IoU threshold. Besides generating a set of predicted boxes, models must also annotate each prediction with a confidence score to rank the items in the curves. These differences encode a different task risk than recall. For example, under this definition a model will be penalized for multiple bounding boxes around one object.

Table 2. Object detection results on COCO before and after reward optimization.

MODEL	MAP (%)	AR@100 (%)
Ours (REWARD$_{\text{MAP}}$)	39.2 → 54.3	54.4 → 67.2
Ours (REWARD$_{\text{RECALL}}$)	N/A	54.4 → 68.4
CHEN ET AL. (2022)	47.1	N/A

One difficulty is that this metric does not decompose into a sum of per-example rewards. We overcome this by noting that mAP metric is well correlated with recall assuming a prediction model does well at ranking the resulting boxes. In order to learn box confidences, we use a supervised loss to learn the expected IoU scores of sampled outputs plus the recall reward defined in the previous section. We additionally improve the reward by computing its value at various IoU ranges (and averaging them) and by weighting each class based on their frequency observed in the training set.

In Table 2 we confirm that by optimizing the proposed reward we drastically improve the mAP score of the original MLE model from 39.2% to 54.3%. In Pix2seq (Chen et al., 2022), the same size ViT-B model with a slightly larger 1333×1333 resolution and many heuristics achieves 47.1%. The best object detection result reported in Pix2seq is 50.0%, when using a larger ViT-L backbone. Our strong ViT-B result clearly demonstrates the promise of the proposed task reward tuning.

By inspecting predictions visually we observe that tuning the proposed reward qualitatively changes the model's predictions: it starts to opportunistically output many more boxes and assigns intuitive confidences to the predicted boxes. It is very different from the behavior of the initial MLE model, which is more conservative in its predictions.

4.3. Colorization

Colorization task is described as adding color to grayscale images. Standard image colorization models are learned by optimizing the likelihood of large datasets of images, i.e. using MLE. Such model generate plausible image coloring, however often produce faded colors. In reality, the user of a colorization model may want to produce a vivid image. Using our approach we demonstrate that MLE colorization models can be tuned to produce colorful images that are more visually appealing.

MLE Pretraining. Similar to the panoptic task, we use UViM colorization as the MLE model. It is a ViT-L/16 encoder for 512×512 resolution images and a 24 layers auto-regressive decoder. The decoder output is a 256 discrete sequence which can then be decoded by UViM stage I models into a 512×512 image. The model was trained using MLE on ImageNet.

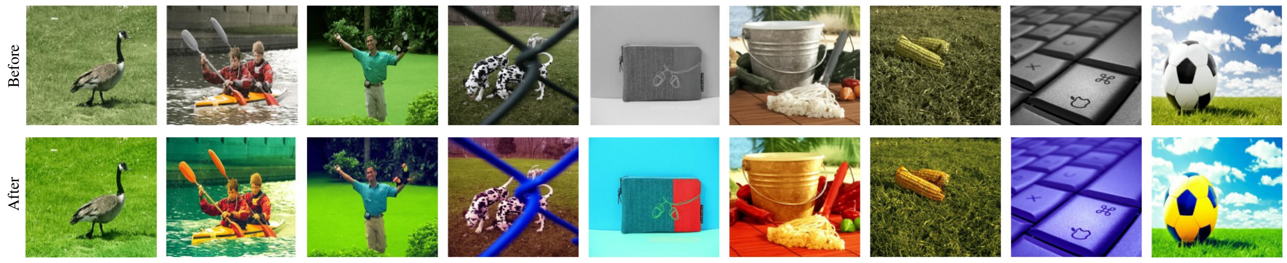


Figure 2. Random examples demonstrating how UViM colorization model (Kolesnikov et al., 2022) predictions change after tuning with a "colorfulness" reward. See text in 4.3 for the details on the reward function.

Tuning for "Colorfulness". We design a custom reward that promotes "colorfulness". In particular, the reward is a product of two terms that are derived from the input image converted to the `Lab` colorspace. In this colorspace, the L channel encodes "lightness", while the a and b channels encode color. The first term of our reward discourages gray colors. It is defined as the fraction of image pixels that have sufficiently "vivid color", where vivid color is defined as $a^2+b^2 > 10$. The second term of our reward promotes color diversity. It is defined as the image-level entropy of the hue value, with hue being computed by $\mathrm{arctan}(\frac{b}{a})$. Note that to compute the entropy we discretize hue into 7 discrete values, distributing the bins uniformly within the range where hue values are defined.

We tune the MLE model with this reward for 1 k steps using a constant learning rate of $3 \cdot 10^{-7}$. As a result of this tuning step, the first reward term grows from 0.46 to 0.97, indicating that the vast majority of predicted colors have become more vivid. The second reward term, hue entropy, grows from 1.03 to 1.84, indicating much greater diversity of predicted colors. We present qualitative results in Figure 2 which clearly demonstrate that the new model consistently produces more colorful images.

Designing the reward for the colorization task was not trivial and we initially had issues with the so-called "reward-hacking" behavior. For example, when directly optimizing for color saturation, the model quickly learned to output extremely saturated colors. Thus, we designed a custom reward that is constant once color saturation surpasses a certain threshold. Also we introduce the color entropy term, as otherwise the model quickly learns to use a single dominant color for the whole image.

4.4. Image Captioning

Image captioning refers to the task of generating textual descriptions for given images. CIDEr (Vedantam et al., 2015) is a popular automated metric that measures caption quality based on consensus with a set of human-written reference captions for the image. Specifically, it measures the n-gram similarity against multiple references captions and takes into account the statistics of the whole dataset such that words that appear more frequently across all captions are given less weight, as they can be considered to be less informative. As mentioned before, the use of REINFORCE to optimize a CIDEr reward is an established technique in image captioning (Rennie et al., 2017). We include it in this work for completeness.

Table 3. Results on COCO caption on Karpathy & Fei-Fei (2015) test split before and after reward optimization.

MODEL	CIDER
Ours (VIT-B)	120.0 → 134.5
Ours (VIT-L)	121.7 → 138.7
WANG ET AL. (2022)	N/A → 138.2
HU ET AL. (2022)	128.7 → 143.7

MLE Pretraining. We pretrain an encoder-decoder Transformer model on COCO captions. We initialize the ViT encoder from the ImageNet21k models provided by Steiner et al. (2021). For the decoder, we randomly initialize a 6-layer auto-regressive decoder. Additionally, we use the BERT (Devlin et al., 2018) 30 k vocabulary to represent text as a discrete sequence of 128 tokens. We pretrain with batch size 256 for 5 k steps using 1 k steps linear warmup followed by cosine schedule with learning rate $3 \cdot 10^{-4}$ and 10x smaller for the encoder parameters. We experiment with two settings: ViT-B/16 and ViT-L/16 both using with the same hyper-parameters.

Tuning for CIDEr. As previous works we use CIDEr directly as reward using the training set to compute the statistics for the n-gram weights. In this case, we use 7 other samples to estimate the reward baseline. We optimize with batch size 256 for 10 k steps with 1 k linear warmup and constant learning rate 10^{-5} with 10x smaller learning rate for the encoder parameters. For reference we include two recent works Wang et al. (2022); Hu et al. (2022) which also utilize CIDEr optimization and recent architectures.

The results in Table 3 show that applying the presented approach results in improvements to the MLE model consistent to observations in prior literature, demonstrating the effectiveness of tuning for a specific task risk. However when inspecting generated captions we do observe the model to output unfinished sentences such as ending with "with a". Those extra n-grams may slightly increase CIDEr, but would significantly decrease the quality of the captions to a human evaluator.

5. Analysis

5.1. Reward Distribution

Here, we analyse rewards of the models in the image captioning example. We compare direct samples from the MLE model (before) to the reward tuned model (after).

We sample 10000 predictions from each model and plot their rewards using the mean of per-example quantile functions in Figure 3a. The area under the curve of this plot gives the expected reward and it shows the clear improvement of the model: over 50% of the tuned model samples have an expected reward higher than 125, however for the MLE model only less than 5% achieves the same standard. Note also, that in the top 1%-tile, the MLE model is capable of generating very high rewards: this indicates that the MLE model is capable of generating high-quality predictions, however, as we cannot select the best samples at test time, we cannot benefit from them.

To demonstrate this, we additionally illustrate, for each example in the dataset, the reward of (1) the prediction with the highest reward out of N samples in Figure 3b and (2) the highest likelihood sample out of N predictions in Figure 3c. We aggregate the cross-dataset average of these two statistics and plot them against the number of samples taken for each example N.

Figure 3b shows that in a large pool of samples, above 100, there exist better samples in the MLE model pool. However to benefit from this fact, one would need an effective strategy to select the best performing sample. Figure 3c shows what happens when we use max likelihood (e.g. greedy, top-k, nucleus sampling) to select the sample: with the increase of the number of samples N, we do see a significant boost in performance for the MLE model inline with greedy/nucleus expectations, however ultimately the performance is much worse than the reward-tuned model even at $N = 10000$.

5.2. Reward-Risk Progression

In order to translate a task risk into a reward function, often we need to decompose a metric computed for a set of examples into per-example reward. This could potentially result in undesirable divergence in the progression of per-example reward and the metric. To see if this is the case empirically, we plot the progression of reward and goal metrics during training in Figure 4 for object detection and panoptic segmentation. We observe no significant divergence between our reward and the metric.

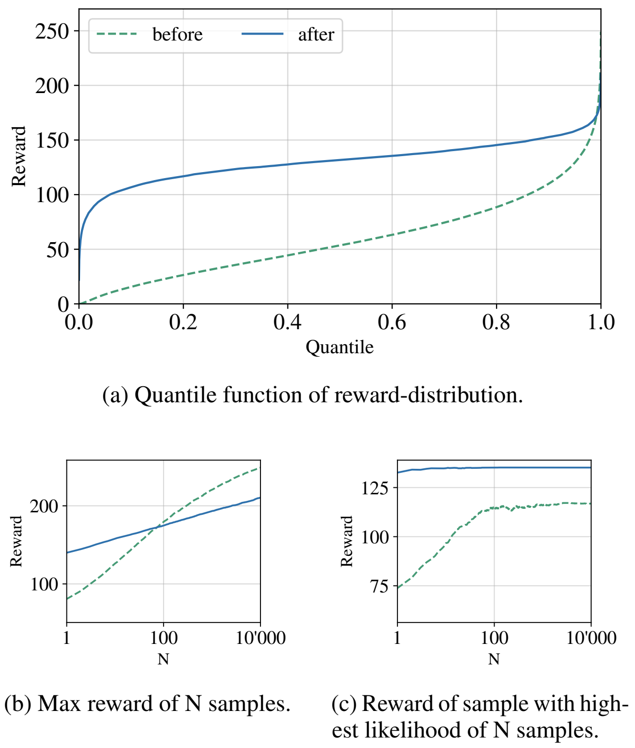
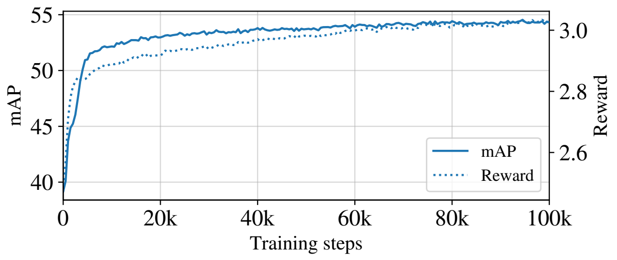


Figure 3. Analysis of reward distribution before and after tuning the model for the image captioning task. Measured as mean of 1024 validation examples. In quantile **plot (a)** the AUC shows the difference between the methods while highlighting that after reward optimization the chance of sampling low-performing samples is greatly reduced. **Plot (b)** the max reward out of N samples shows that the MLE model includes high-quality outputs in a large enough pool. **Plot (c)** it is not possible to identify the best outputs using likelihood as observed by the low reward when using the most likely out of N samples even when using 10000 samples.

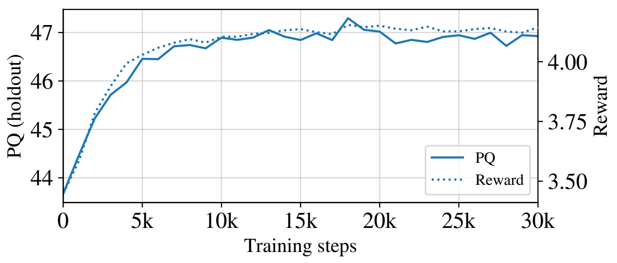
Additionally, in Figure 4 (a), we observe that the mAP score of object detection quickly goes up for the first $20k$ steps, i.e. from 39.2% to 52.3%. With longer reward tuning, the metric keeps going up and it achieves 52.7% at 40 k steps and 53.2% at 60 k steps.

5.3. Ablation of Reward Baseline

To ablate the need of a reward baseline and the impact of using more samples to compute it, we perform additional experiments for panoptic segmentation using ViT-L and training for 10 k steps and captioning using ViT-B and training for 10 k steps. Results in Table 4 show there is a clear gain from using a baseline, but the gain does not improve significantly when increasing the number of samples.



(a) Object detection.



(b) Panoptic segmentation.

Figure 4. Plot of metrics and rewards measured on the validation set during training for object detection (mAP) and panoptic segmentation (PQ). Overall we observe a good correlation between the reward being optimized and the task risk.

Table 4. Results on panoptic segmentation (PQ) and image captioning (CIDEr) tasks with varying number of samples used to compute the baseline.

TASK	MLE	NO BASELINE	BASELINE SAMPLES 1	3	7
PANOPTIC	43.1	44.6	46.0	46.1	46.2
CAPTION	120.0	124.2	132.9	134.2	134.3

6. Discussion and Limitations

Reward Hacking. Our work shows the feasibility of tuning a generic model to a task by using rewards. This simplifies and opens up the possibilities of training models to new tasks, in particular to tasks where a reward can be defined by using real-risk or human-feedback. However, it is important to note that there is no guarantee that a given reward function will result in improvements in the intended usage. The model may instead exploit weaknesses in the reward definition as we briefly discuss in the colorization and captioning task. It is crucial to take that in consideration when validating the models.

Reward Design. In this work we mostly use simple rewards based on existing evaluation metrics. There are however many more options, including combinations of filter-based, input-output checks, simulation checks, use of pretrained models to detect undesired outputs or to keep the model closer to a initial distribution, entropy to encourage diversity and exploration or use of real-world or human feedback. Another thing to keep in mind is that the reward does not needs to aim to be as exact as the task risk. Alternatives functions may be easier to obtain, control or optimize, for example if the reward provides more guidance than a sparse risk value. We leave the exploration of this in computer vision tasks to future work.

Advanced RL Techniques. The presented approach with warm-up and constant learning rate setup suffices across the explored applications. As such, we saw no need to add regularisation to remain close to the original policy, encourage exploration or attempt to reduce the number of reward calls or model updates. We believe the efficacy is in part due to the MLE-pretrained initialization, allowing the model to avoid potential issues with the simple technique. Although this may not hold in more complex setups, we encourage to try the simple approach in other similar applications.

Data for Imitation Learning. Can MLE-training alone also reach better alignment with the task goal? This question was part of the motivation of this work. Although we expect more data to help MLE models imitate the ground truth, we found it hard to know what data to collect or how to augment it to have a particular alignment effect with a task risk. By tuning a model with a reward we obtain that effect by optimizing a model to avoid undesired outputs. Since the space of undesired outputs where the MLE model assigns high likelihood is hard to predict, it is critical to observe the model in action and focus on the examples the model misassigns high-likelihood.

Training Cost. There are two main costs to consider: model sampling cost and the number of queries to the reward function. Sampling auto-regressive models is notoriously more expensive than computing the likelihood of a given sequence. This is due to difficulties utilizing hardware efficiently and not due to an increase in the number of flops. Note however that this cost is still proportional to inference usage. Additionally, the presented method only requires a model where the likelihoods of samples can be optimized with gradients. It does not depend on the model being auto-regressive, though that can be an important piece to modelling complex distributions. For more complex applications the number of queries to the reward function can be a bigger concern. In such cases it is worth to explore off-policy RL techniques and approximate a target reward with a value network.

7. Conclusion

Our work shows that reward optimization is a viable option to optimize a variety of computer vision tasks. Using the simple approach of pretraining to imitate ground truth followed by reward optimization, we were able to: (a) improve models for object detection and panoptic segmentation trained without other task-specific components to the level comparable to ones obtained through clever manipulation of data, architectures and losses; (b) qualitatively affect the results of colorization models to align with a goal of creating vivid and colorful images; (c) show that the simple approach is competitive with recent works in captioning.

We believe these results demonstrate the possibilities to have more precise control on how models can be aligned with non-trivial task risks. Opening up the use cases for general modelling approaches and their abilities to tackle more challenging use cases. We look forward to follow-up research such as tuning scene understanding outputs for robot grasping, where one can optimize the perception models for the probability of a successful grasp.

Acknowledgements

We thank Jeremiah Harmsen, Maxim Neumann, Olivier Bachem, Xi Chen and Neil Houlsby for discussions during this project. In addition we thank Jeremiah Harmsen and Matthieu Geist for feedback on the text of this paper. We use the `big_vision` codebase (Beyer et al., 2022) for all experiments in this project.

References

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. *NeurIPS*, 2015.

Beyer, L., Zhai, X., and Kolesnikov, A. Big Vision. `https://github.com/google-research/big_vision`, 2022.

Bhowmik, A., Gumhold, S., Rother, C., and Brachmann, E. Reinforced feature points: Optimizing feature detection and description for a high-level task. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *ECCV*, 2020.

Chen, T., Saxena, S., Li, L., Fleet, D. J., and Hinton, G. Pix2seq: A language modeling framework for object detection. In *ICLR*, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint:1810.04805*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Glaese, A., McAleese, N., Trebacz, M., Aslanides, J., Firoiu, V., Ewalds, T., Rauh, M., Weidinger, L., Chadwick, M., Thacker, P., et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint:2209.14375*, 2022.

Henderson, P. and Ferrari, V. End-to-end training of object class detectors for mean average precision. In *ACCV*, 2017.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. In *ICLR*, 2020.

Hu, J. C., Cavicchioli, R., and Capotondi, A. Expansionnet v2: Block static expansion in fast end to end training for image captioning. *arXiv preprint:2208.06551*, 2022.

Huang, C., Zhai, S., Guo, P., and Susskind, J. Metricopt: Learning to optimize black-box evaluation metrics. In *CVPR*, 2021.

Karpathy, A. and Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

Keneshloo, Y., Shi, T., Ramakrishnan, N., and Reddy, C. K. Deep reinforcement learning for sequence-to-sequence models. *IEEE TNNLS*, 31(7):2469–2489, 2019.

Kirillov, A., He, K., Girshick, R., Rother, C., and Dollar, P. Panoptic segmentation. In *CVPR*, 2019.

Kolesnikov, A., Guillaumin, M., Ferrari, V., and Lampert, C. H. Closed-form training of conditional random fields for large scale image segmentation. *ECCV*, 2014.

Kolesnikov, A., Pinto, A. S., Beyer, L., Zhai, X., Harmsen, J. J., and Houlsby, N. UVim: A unified modeling approach for vision with learned guiding codes. In *NeurIPS*, 2022.

Krähenbühl, P. and Koltun, V. Efficient inference in fully connected crfs with gaussian edge potentials. *NeurIPS*, 2011.

Kreutzer, J., Khadivi, S., Matusov, E., and Riezler, S. Can neural machine translation be improved with user feedback? In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (Industry Papers)*, June 2018.

Krull, A., Brachmann, E., Nowozin, S., Michel, F., Shotton, J., and Rother, C. Poseagent: Budget-constrained 6d object pose estimation via reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Lafferty, J., McCallum, A., and Pereira, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

Le, N., Rathour, V. S., Yamazaki, K., Luu, K., and Savvides, M. Deep reinforcement learning in computer vision: a comprehensive survey. *Artificial Intelligence Review*, 2021.

Leblond, R., Alayrac, J.-B., Sifre, L., Pislar, M., Jean-Baptiste, L., Antonoglou, I., Simonyan, K., and Vinyals, O. Machine translation decoding beyond beam search. In *EMNLP*, 2021.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *ICCV*, 2017.

Mathe, S., Pirinen, A., and Sminchisescu, C. Reinforcement learning for visual object detection. In *CVPR*, 2016.

Nowozin, S., Rother, C., Bagon, S., Sharp, T., Yao, B., and Kohli, P. Decision tree fields. In *ICCV*, 2011.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Gray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *ICML*, 2021.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. Sequence level training with recurrent neural networks. *arXiv preprint:1511.06732*, 2015.

Rao, Y., Lin, D., Lu, J., and Zhou, J. Learning globally optimized object detector via policy gradient. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6190–6198, 2018.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 28, 2015.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. Self-critical sequence training for image captioning. In *CVPR*, 2017.

Schmidt, F. Generalization in generation: A closer look at exposure bias. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, 2019.

Shao, S., Li, Z., Zhang, T., Peng, C., Yu, G., Zhang, X., Li, J., and Sun, J. Objects365: A large-scale, high-quality dataset for object detection. In *ICCV*, 2019.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In *ICML*, 2018.

Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. Minimum risk training for neural machine translation. *arXiv preprint:1512.02433*, 2015.

Song, Y., Schwing, A., Urtasun, R., et al. Training deep neural networks via direct loss minimization. In *ICML*, 2016.

Stahlberg, F. and Byrne, B. On NMT search errors and model errors: Cat got your tongue? In *EMNLP-IJCNLP*, 2019.

Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., and Beyer, L. How to train your ViT? data, augmentation, and regularization in vision transformers. *arXiv preprint:2106.10270*, 2021.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *NeurIPS*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Vedantam, R., Lawrence Zitnick, C., and Parikh, D. Cider: Consensus-based image description evaluation. In *CVPR*, 2015.

Wang, Y., Xu, J., and Sun, Y. End-to-end transformer based model for image captioning. *arXiv preprint:2203.15350*, 2022.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. Scaling vision transformers. In *CVPR*, 2022.

A. Overview of models and hyper-parameters

Table 5. Panoptic segmentation settings.

Model	
Encoder:	ViT-L/16
Decoder:	24 layers
Seq length:	256 tokens
Resolution:	512×512
MLE pretraining	
Model	Kolesnikov et al. (2022)
Tune for PQ	
Batch size:	128
Schedule:	constant
Learning-rate:	$1 \cdot 10^{-6}$
Total steps:	30 000
Warmup steps:	4 000

Table 6. Object detection settings.

Model	
Encoder:	ViT-B/16
Decoder:	6 layers
Seq length:	600 tokens
Resolution:	1280×1280
MLE - Objects365 pretraining	
Resolution:	640×640
Batch size:	256
Learning-rate:	$1 \cdot 10^{-3}$
Weight-decay:	$5 \cdot 10^{-5}$
Schedule:	cosine
Total steps:	400 000
Warmup steps:	20 000
MLE - COCO finetune	
Resolution:	1280×1280
Batch size:	256
Learning-rate:	$1 \cdot 10^{-4}$
Weight-decay:	0.0
Schedule:	cosine
Total steps:	10 000
Warmup steps:	1 000
Tune for recall	
Batch size:	256
Learning-rate:	$1 \cdot 10^{-6}$
Schedule:	constant
Total steps:	100 000
Warmup steps:	1 000
Tune for mAP	
Batch size:	256
Learning-rate:	$1 \cdot 10^{-6}$
Schedule:	constant
Total steps:	100 000
Warmup steps:	1 000

Table 7. Colorization settings.

Model	
Encoder:	ViT-L/16
Decoder:	24 layers
Seq length:	256 tokens
Resolution:	512×512
MLE pretraining	
Model	Kolesnikov et al. (2022)
Tune for "colorfulness"	
Batch size:	512
Schedule:	constant
Learning-rate:	$3 \cdot 10^{-7}$
Total steps:	1 000
Warmup steps:	0

Table 8. Image captioning settings.

Model	
Encoder:	ViT-B/16 / ViT-L/16
Decoder:	6 layers
Seq length:	128 tokens
Resolution:	512×512
MLE pretraining	
Encoder ckpt:	ImageNet21k from Steiner et al. (2021)
Batch size:	256
Decoder lr:	$3 \cdot 10^{-4}$
Encoder lr:	$3 \cdot 10^{-5}$
Weight-decay:	$5 \cdot 10^{-6}$
Schedule:	cosine
Total steps:	5 000
Warmup steps:	1 000
Tune for CIDEr	
Batch size:	256
Baseline:	7 samples
Decoder lr:	$1 \cdot 10^{-5}$
Encoder lr:	$1 \cdot 10^{-6}$
Schedule:	constant
Total steps:	10 000
Warmup steps:	1 000