MTAdam: Automatic Balancing of Multiple Training Loss Terms

Itzik Malkiel
Tel Aviv University
itzik.malkiel@gmail.com

Lior Wolf
Tel Aviv University
wolf@cs.tau.ac.il

Abstract

When training neural models, it is common to combine multiple loss terms. The balancing of these terms requires considerable human effort and is computationally demanding. Moreover, the optimal trade-off between the loss terms can change as training progresses, e.g., for adversarial terms. In this work, we generalize the Adam optimization algorithm to handle multiple loss terms. The guiding principle is that for every layer, the gradient magnitude of the terms should be balanced. To this end, the Multi-Term Adam (MTAdam) computes the derivative of each loss term separately, infers the first and second moments per parameter and loss term, and calculates a first moment for the magnitude per layer of the gradients arising from each loss. This magnitude is used to continuously balance the gradients across all layers, in a manner that both varies from one layer to the next and dynamically changes over time. Our results show that training with the new method leads to fast recovery from suboptimal initial loss weighting and to training outcomes that match or improve conventional training with the prescribed hyperparameters of each method.

1 Introduction

In both supervised and unsupervised learning, adding loss terms often leads to improved performance. However, as more loss terms are added, the space of possible balancing weights increases exponentially, and more resources need to be allocated to identify good configurations that would justify the added terms. Another common challenge is that at different stages of the training process, the optimal balance may change (Mescheder et al., 2017). It is, therefore, necessary to have the balancing terms update dynamically during training, which further increases the hyperparameter space.

In this work, we introduce Multi-Term Adam (MTAdam), an optimization algorithm for multi-term loss functions. MTAdam extends Adam (Kingma and Ba, 2014) and allows an effective training of an unweighted multi-term loss objective. Thus, MTAdam can streamline the computationally demanding task of hyperparameter search, which is required for effectively weighting multi-term loss objectives.

At every training iteration, a dynamic weight is assigned to each of the loss terms, based on the magnitude of the gradient that each term entails. The weights are assigned in a way that balances between these gradients and equates their magnitude.

This, however, would be an ineffective balancing method, without two crucial components: (i) the balancing needs to occur independently for each layer of the neural network and separately over the losses, since the relative contributions of the losses vary, depending on the layer, and (ii) the update step needs to take into account the maximal variance among all losses, to support sufficient explorations in places of the parameter space, in which one of the losses becomes more sensitive.

The main focus of our experiments is in the domains of natural language understanding and conditional adversarial image generation, in which the leading methods often utilize multiple loss terms. Our results show that MTAdam is able to recover from suboptimal starting points, in which the weight parameters are set inappropriately, while Adam and other baseline methods cannot.

2 Related Work

SGD with momentum (Nesterov) (Rumelhart et al., 1986) is an optimization algorithm that extends SGD, suggesting to update the network's parameters by a moving average of the gradients, rather than the gradients at each step. Root Mean Square Propagation (RMSProp) (Tieleman and Hinton, 2012) extends SGD, dividing the learning rate during the backward step, by the moving average of the second moment of the gradients of each parameter.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 10713–10729
November 7–11, 2021.

Adam (Kingma and Ba, 2014) combines both principles and employs the first and second moments of the gradients of each learned parameter and applies them during the backward step. Adam has become a dominant optimizer, that is applied across many applications, and, in particular, it is the de-facto standard in the field of adversarial training. It is known for improving convergence to work well with the default values of its own hyperparameters.

Multiple methods have been suggested for selecting hyperparameters (Srinivas et al., 2009; Wang et al., 2016; Bergstra et al., 2011; Feurer et al., 2014). Hyperband (Li et al., 2017) performs hyperparameter search as an infinite-armed bandit problem utilizing a predefined amount of resources, while searching for the best configuration of hyperparameters that maximizes the given success criterion. It can provide an order-of-magnitude speedup compared to Bayesian optimization (Bergstra et al., 2011; Snoek et al., 2012; Hutter et al., 2011).

In our experiments, we employ multiple methods from natural language understanding and the field of image generation. The methods vary in the type of supervision employed or the task being solved: (i) BERT (Devlin et al., 2019) is a pre-trained transformer-based (Vaswani et al., 2017) language model, that applies fine-tuning on various language understanding tasks using supervision. (ii) RecoBERT (Malkiel et al., 2020a) is a text-similarity model that fine-tunes BERT on a given catalog, by jointly optimizing a standard masked language modeling, along with a unique title-description model. (iii) pix2pix (Isola et al., 2017) generates an image in domain B based on an input image in domain A, after observing matching pairs during training. (iv) CycleGAN (Zhu et al., 2017) performs the same task, while training in an unsupervised manner on unmatched images from the two domains. (v) SRGAN (Ledig et al., 2017a) generates high-resolution images from low-resolution ones and is trained in a supervised way.

3 Motivation: Multi-loss Dynamics

To examine the challenges that arise when not scaling the losses correctly, we consider a simple regression problem that involves the L1 or the L2 loss. Specifically, we fit two coefficients to learn the polynomial $f(x) = ax + bx^2$. To this end, we randomly sample batches of $x \in \mathbb{R}$ from a normal distribution, and train a network to minimize the loss, i.e, the network performs a dot product between (x, x^2) and the vector of parameters $W \in \mathbb{R}^2$, and the objective compares between $(x, x^2)W$ and $f(x)$: for L1 $|(x, x^2)W - f(x)|$ and for L2 $\|(x, x^2)W - f(x)\|_2^2$.

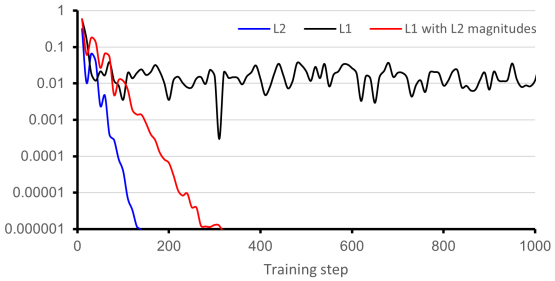


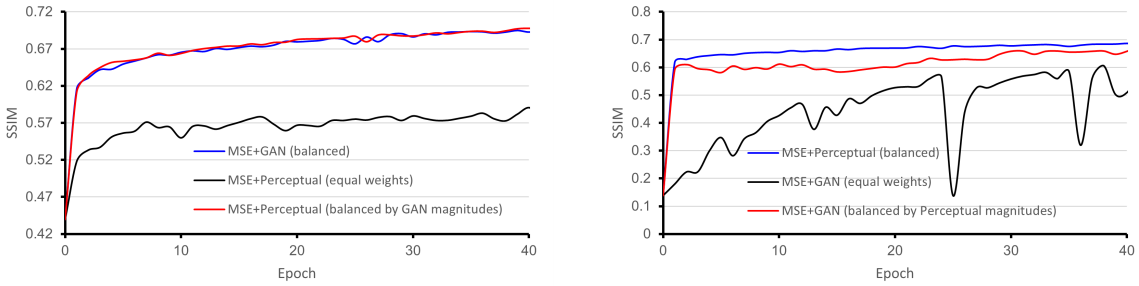
Figure 1: Learning the coefficients of $f(x) = ax + bx^2$, minimizing L1, L2, or L1 where the gradients are scaled with the magnitudes of the L2 gradients (training step vs. loss).

We track and analyze the loss and its gradients on the two parameters. We run three Adam experiments with the values $a = 1.52$, $b = -0.29$ (picked at random, results are very similar for other values). Each time we train the model with a different objective: L1, L2, and L1 where the gradients are scaled with the magnitudes of the L2 gradients. The latter utilizes the L1 loss and scales the gradient magnitudes of the coefficients to match the magnitudes created by the L2 loss, i.e. we apply $(\frac{\partial L1}{\partial W_1}, \frac{\partial L1}{\partial W_2}) := (\frac{\partial L1}{\partial W_1}, \frac{\partial L1}{\partial W_2}) \frac{\|(\frac{\partial L2}{\partial W_1}, \frac{\partial L2}{\partial W_2})\|_2}{\|(\frac{\partial L1}{\partial W_1}, \frac{\partial L1}{\partial W_2})\|_2}$ during the backward propagation step.

As shown in Fig. 1 (log scale), the L2 loss converges to a lower loss than L1, since the latter oscillates around 10^{-2}. The underlying reason is that the gradients' magnitude of the two coefficients w.r.t. the L2 loss constantly decreases during training, while the gradients' magnitude w.r.t. the L1 remains unchanged (see appendix for a plot). Let $t := (x, x^2)W - f(x)$, the gradient of $|t|$ w.r.t. t are $\frac{\partial}{\partial t}|t| = 1$ if $t > 0$ and $\frac{\partial}{\partial t}|t| = -1$ if $t < 0$. In contrast, the L2 gradients decrease with model improvements, since $\frac{\partial}{\partial t}t^2 = 2t$.

Fig. 1 also shows that when scaled by the L2 gradient magnitude, the L1 training converges to the optimal solution. We, therefore, observe that the gradient magnitude of a loss that converges well, can scale the gradients of a loss which converges to a larger error. This observation implies that instead of finding a training schedule for a new loss, we can rely on an existing loss for scaling.

We further explore this in a scenario with multiple loss terms. It is often the case that new loss



(a) (b)

Figure 2: We train a superresolution image mapping technique using two out of three loss terms: MSE, GAN, and perceptual loss. (a) the GAN loss is well-balanced and we obtain effective training when combining it with the MSE loss (blue curve). However, the weight of the perceptual loss was naively set to 1, which is two orders of magnitude larger than the optimal value for combining it with the MSE loss. This leads to poor SSIM (Wang et al., 2004) scores (black curve). However, when scaling the perceptual loss gradients by the magnitude of the gradients of the GAN loss, the perceptual loss supports the MSE loss (red curve). (b) same, where the roles of the perceptual loss and the GAN loss are reversed. Similar FID (Heusel et al., 2017) graphs can be found in the appendix.

terms are added during training. However, one would like to avoid tuning each one separately. As a computationally efficient sample, we consider supervised image superresolution using two out of the three loss terms of Ledig et al. (2017b): MSE, a GAN which verifies that the result is in the target domain, and a perceptual loss. We consider the case in which either the GAN loss or the perceptual loss are weighted equally, in comparison to the MSE (a weight that is larger by two orders of magnitude than the optimal) and observe, in Fig. 2 that this leads to suboptimal results. However, using the gradient magnitude of a balanced term fixes this and allows the method to train well. Thus, this motivating experiment demonstrates that we can balance a loss term by using the gradient magnitude of another term. Experiments with the three terms together are shown in Sec. 5.

4 Method

The Adam algorithm optimized one stochastic objective function $f_t(\theta)$ over the set of parameters θ, where t is an index of the current mini-batch of samples. In contrast, MTAdam optimizes a set of such terms $f_t^1(\theta), \ldots, f_t^I(\theta)$. While Adam's task is to minimize the expected value $E_t[f_t(\theta)]$ w.r.t. the parameters θ, MTAdam minimizes a weighted average of the I terms. The weights of these mixtures are all positive, but otherwise unknown. The guiding principle for the determination of the weights at each iteration t is that the moving average of the magnitude of the gradient of each term is equal across terms. This magnitude is evaluated and balanced at every layer of the network.

In Adam, two moments are continuously updated, using a moving average scheme: m_t is the first moment of the gradient $\nabla_\theta f_t$ and v_t is the second moment. Both are vectors of the same size of θ. The moving averages are computed using the mixing coefficients β_1 and β_2 for the two moments.

MTAdam records such moments for each term $i = 1...I$ separately. In addition, it uses a mixing coefficient β_3 in order to maintain the moving average of the gradient magnitude per each layer ℓ, which is denoted by $n^i_{l,t}$.

Adam borrows from the SGD with momentum method (Nesterov) and updates the vector of parameters based on the weighted first moment of the gradient. In MTAdam, the first moment is computed based on a weighted gradient, in which the parameters of each layer ℓ for every term i are weighted such that their magnitude is normalized by the factor $n^i_{\ell,t}$. This way, across all layers, and at every time point, the I terms contribute equally to the gradient step.

The Adam algorithm is depicted in the left side of Alg. 1 and MTAdam on the right. In line 1, MTAdam initializes I pairs of first and second moment vectors. This is similar to Adam, except for initializing a pair of moments for each loss term. In line 2, and different from Adam, MTAdam initializes I first moments for the magnitude of the gradients, per layer. In line 3, both MTAdam and Adam iterate over the stochastic mini-batches, performing T training steps.

In line 4, MTAdam iterates over the loss terms.

Algorithm 1: Adam (left) and Multi-term Adam (right). All operations are element-wise.

Input: α: step size, $\{\beta_1, \beta_2\}$: decay rates to calculate the 1st and 2nd moments, θ_0: initial weights, $f_t(\theta)$: stochastic objective.
Output: θ_t: resulting parameters

$m_0, v_0 \leftarrow 0, 0$

while $t = \{1, \cdots, T\}$ **do**

  $g \leftarrow \nabla_\theta f_t(\theta_{t-1})$

  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g$
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g^2$
  $\widehat{m_t} \leftarrow m_t / \left(1-\beta_1^t\right)$
  $\widehat{v_t} \leftarrow v_t / \left(1-\beta_2^t\right)$

  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m_t} / (\sqrt{\widehat{v_t}} + \epsilon)$

return θ_T

Input: In addition to Adam's parameters: β_3: decay rate for the gradient's norm first moment, $f_t^1(\theta)...f_t^I(\theta)$: stochastic loss term functions.
Output: θ_t: resulting parameters

1 **for** $i = \{1, \cdots, I\}$ **do** $m_0^i, v_0^i \leftarrow 0, 0$
2 **for** l *in* $1...L$ **do** $n_{\ell,0}^i \leftarrow 1$
3 **while** $t = \{1, \cdots, T\}$ **do**
4 **for** $i = \{1, \cdots, I\}$ **do**
5 $g^i := (g_1^i, ...g_L^i) \leftarrow \nabla_\theta f_t^i(\theta_{t-1})$
6 **for** ℓ *in* $1...L$ **do**
7 $n_{\ell,t}^i \leftarrow \beta_3 \cdot n_{\ell,t-1}^i + (1-\beta_3) \cdot \|g_\ell^i\|_2$
8 $g_\ell^i \leftarrow n_{\ell,t}^1 \cdot g_\ell^i / (n_{\ell,t}^i)$
9 $m_t^i \leftarrow \beta_1 \cdot m_{t-1}^i + (1-\beta_1) \cdot g^i$
10 $v_t^i \leftarrow \beta_2 \cdot v_{t-1}^i + (1-\beta_2) \cdot (g^i)^2$
11 $\widehat{m_t^i} \leftarrow m_t^i / \left(1-\beta_1^t\right)$
12 $\widehat{v_t^i} \leftarrow v_t^i / \left(1-\beta_2^t\right)$
13 **for** $i = \{1, \cdots, I\}$ **do**
14 $\theta_{t-1} \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m_t^i} / (\sqrt{\max(\widehat{v_t^1}...\widehat{v_t^I})} + \epsilon)$
15 $\theta_t \leftarrow \theta_{t-1}$
16 **return** θ_T

For each term, MTAdam calculates its gradients over each one of the layers (line 5), analogously to the way Adam computes the (single) gradients vector $\nabla_\theta f_t(\theta_t - 1)$. In lines 6-8, MTAdam iterates over the layers, updates the moving average of the magnitude for each layer and loss , and normalizes the gradients of the current layer and loss term, by multiplying with $\frac{n_{\ell,t}^1}{n_{\ell,t}^i}$. This multiplication normalizes the magnitude of the current gradients of layer ℓ and loss term i using the moving average n_ℓ^i. This normalization leads to all gradient magnitudes to be similar to that of the first loss term. This assigns a unique role to the first term, as the primary loss to which all other losses are compared. By linking the magnitude to that of a concrete loss, and not to a static value (e.g., normalizing to have a unit norm), we maintain the relationship between the training progression and the learning rate.

While line 5 has an analog in Adam, in MTAdam, the computation of the gradients of each specific loss term i w.r.t θ is done per layer. For a layer index ℓ, the gradient is denoted by g_ℓ^i. We denoted by g^i the concatenation of all per-layer gradients. Lines 6-8 do not have an Adam analog.

The normalization iterates over the loss terms, and for each gradient, the first moment of the magnitude of the gradient is updated. The gradient magnitude is then normalized by that of the first loss term.

Then, in lines 9-12, MTAdam updates the first and second moments for each parameter and each loss term and computes their bias correction. This is similar to Adam, except that the moments are calculated separately for each loss term. In lines 13-15, MTAdam iterates over the loss terms and calculates the steps from each term. The steps are summed over θ_{t-1}, and the result is assigned θ_t.

In Adam, the update size is normalized by the second moment. In MTAdam, we divided by the maximal second moment among all loss terms. This division allows MTAdam to make smaller gradient steps, when a lower certainty is introduced by at least one of the loss terms. The motivation for this is that even if one of the losses is in a high-sensitivity region, where small updates create rapid changes to this term, then the step, regardless of the term which led to it, should be small. The importance of this maximization is demonstrated in the ablation study in Sec. 5.5.

Memory and Run Time Analysis Adam utilizes a pair of 1st and 2nd moments for each learned parameter. Given a network with $\|\theta\|$ learned parameters, it has a memory complexity of $O(|\theta|)$. MTAdam utilizes I different pairs of 1st and 2nd moments for each parameter and also I first moments magnitude for each layer. These two extensions bring the memory complexity to $O(I|\theta| + IL) \sim O(I|\theta|)$. The run time complexity of Adam per each training step is linear in the number of parameters $O(\theta)$. In MTAdam, the run

Suboptimal weighting				Optimal
SGD-Momentum	RMSProp	Adam	MTAdam	Adam
85.4 ± 1.2	87.2 ± 1.0	88.8 ± 0.8	97.9 ± 0.1	98.3 ± 0.1

Table 1: Mean accuracy $\pm$ SD on MNIST (100 runs).

time complexity also depends on the number of loss terms $O(I\theta)$. The dependence on the number of layers L in Alg. 1 can be absorbed in θ.

5 Experiments

We compare the results of MTAdam with five baselines: (1-3) Adam, RMSProp, and SGD with momentum, applied with suboptimal weightings. (4) Hyperband (Li et al., 2017) applied to perform a hyperparameter search for the weighting parameters (lambdas). (5) Adam optimizer applied with the prescribed weighting.

We note that baseline (4) benefits from running training multiple times and that baseline (5) employs the weights proposed for each method after a development process that is likely to have included a hyper-parameter search, in which multiple runs were evaluated by the developers of each method. For each optimization method, we employ the default parameters in pytorch. For all Adam and MTAdam experiments, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For MTAdam $\beta_3 = 0.9$.

5.1 MNIST classification

In order to turn MNIST into a suboptimal combination of multiple terms experiment, we compute the loss for each of the digits separately, creating ten losses, each weighted by a random weight from a uniform distribution between 1 and 1000. The test set is unweighted, which causes classes associated with lower weights to suffer from underfitting.

The official of the PyTorch MNIST example is used: two convolutional layers, followed by two fully connected layers. The experiment is repeated 100 times, and for the sake of saving computations, hyperband is not tested. The results in Tab. 1, show a clear advantage for MTAdam over the other suboptimal alternatives.

5.2 BERT Experiments

To demonstrate the effectiveness of MTAdam to recover from non-optimal weighting when applied to Transformers (Vaswani et al., 2017), we fine-tune BERT (Devlin et al., 2019) with a sub-optimal combination of terms. BERT is commonly applied with AdamW (Loshchilov and Hutter, 2017), which employs an L2 regularization on the network weights. Thus, the BERT objective can be formulated as a dual-loss objective consisting of a standard classification loss and an L2 regularization: $\mathcal{L}_{BERT} = \lambda_1 \mathcal{L}_{bce} + \lambda_2 \mathcal{L}_{reg}$, where $\mathcal{L}_{bce}$ is the binary cross-entropy loss and $\mathcal{L}_{reg}$ is the L2 regularization loss. The approximately optimal λ_1 and λ_2 weights proposed by BERT (Devlin et al., 2019) are 1 and 0.01, respectively.

In our experiments, we fine-tune BERT with MTAdam, minimizing the above objective, where the regularization is applied with equal weighting, setting $\lambda_2 = 1$. We compare MTAdam with Adam applying the same dual-term objective and with the AdamW. Note that AdamW applies the L2 regularization directly on the model weights, i.e., the L2 gradients are not combined with the first and second moments of the main loss.

Performance is evaluated on the MRPC (Dolan and Brockett, 2005), RTE (Bentivogli et al., 2009), and STS-B (Cer et al., 2017), and reported over five experiments for each variant. The results in Tab. 2, shows that MTAdam is the only method to overcome the equal weights, achieving the results obtained by Adam and AdamW on the prescribed weights. The runtime of MTAdam applied on the BERT architecture is comparable to the Adam and AdamW alternatives, with up to 20% additional compute time (measured on a single v100 GPU).

5.3 RecoBERT Experiments

We next apply MTAdam based training to RecoBERT (Malkiel et al., 2020a,b), which is a recent state-of-the-art text-based item similarity model. In the RecoBERT settings, a catalog of textual items is given, where each item is composed of a title-description pair. Item titles are sentences and descriptions are paragraphs. The RecoBERT model is a BERT-based network trained with self-supervision to jointly optimize a standard masked language model (MLM) and a unique title-description model (TDM). During training, title-description pairs are tokenized and masked (similar to BERT), and then propagated through the model. The optimization goal is to reconstruct the masked words, and predict whether a given pair of title and description corresponds to the same item or not. The RecoBERT objective can be expressed as:

$$\mathcal{L}_{\text{RecoBERT}} = \mathcal{L}_{MLM} + \lambda \mathcal{L}_{TDM} \qquad (1)$$

Dataset	no reg.	Equal weighting					Prescribed weights	
	Adam	SGD-Momentum	RMSProp	Adam	AdamW	MTAdam	Adam	AdamW
MRPC	85.9 ± 3.0	68.3 ± 0	68.3 ± 0	68.3 ± 0	86.9 ± 1.1	**87.2** ± 1.1	**87.2** ± 1.3	86.9 ± 1.1
RTE	68.5 ± 11.5	60.5 ± 1.2	54 ± 2.7	55.5 ± 2.8	72.2 ± 1.9	**73.7** ± 1.8	70.0 ± 3.0	69.6 ± 4.5
STS-B	90.1 ± 0.4	80.7 ± 1.3	80.2 ± 1.4	81.3 ± 1.0	89.5 ± 0.1	**90.4** ± 0.2	**90.4** ± 0.1	**90.4** ± 0.2

Table 2: Mean±SD over five runs of Accuracy for MRPC, RTE and STS-B.

where the $\mathcal{L}_{MLM}$ is a standard masked language model loss, λ was set to 1, and $\mathcal{L}_{TDM}$ is the TDM loss, which is a standard cross entropy employed over the cosine between the averaged pooled embeddings of the title and description tokens.

In (Malkiel et al., 2020a), the authors introduced a wines similarity task[1] with a test set of items with similarity annotations crafted by a human expert[2] (see Fig.3).

We evaluate RecoBERT with MTAdam on the wines similarity task and compare its performance to the baselines mentioned above. The Hyperband experiments utilize the validation loss to perform a hyperparameter search on the λ and incorporate 40 trials (i.e. 40 training processes, which gives it a great advantage), each trial randomly sampled a different lambda from the logarithmic scale between $[10^{-4}, 10^{4}]$. We report Hyperband performance, by utilizing the trial that is associated with the chosen lambda ($\lambda = 17.4$).

As can be seen in Tab. 3, RecoBERT applied with MTAdam yields a sizable improvement compared to the original work, and other alternatives. Specifically, compared to the best performing model, MTAdam improves the Mean Percentile Ranking (MPR) score by 0.8, Mean Reciprocal Rank (MRR) by 1.4 and the Hit Rate at 10 (HR@10) by 4.3.

Runtime Comparing the training time for both the BERT and RecoBERT experiments, MTAdam entails an additional $\sim 20\%$ of compute time over Adam. Specifically, in RecoBERT the original model was trained for three days on a single V100 GPU, while MTAdam training consumed an additional 14 hours. Note that no attempt was done yet to optimize the MTAdam code. The hyperband experiments required 16 days (four v100 GPUs running in parallel for 4 days). A grid search of even a coarse resolution would have required at least two orders of magnitude more resources than MTAdam.

[1]dataset: https://www.kaggle.com/zynicide/wine-review
[2]https://doi.org/10.5281/zenodo.3653403

		MPR↑	MRR↑	HR@10↑
equal wei.	SGD-Momentum	95.5%	90.1%	60.8%
	RMSProp	95.8%	91.0%	61.4%
	Adam-prescribed	96.3%	91.7%	65.4%
	MTAdam	**97.4%**	**93.3%**	**70.5%**
	Hyperband-FID	96.6%	91.9%	66.2%

Table 3: RecoBERT results.

Title: Domaines Devillard 2016 Domaine des Perdrix La Perriere Premier Cru (Gevrey-Chambertin)

Review: From just over half an acre, this wine has considerable power. Already balanced, it has intense tannins backed by concentrated black fruits. A strong mineral element partners with juicy acidity. Drink from 2023.

Title: Maison Stephane Brocard 2015 Closerie des Alisiers Les Corbeaux Premier Cru (Gevrey-Chambertin)

Review: This powerful wine is both dense and rich. Its tannins and acidity bode well for aging, while the red plum and berry fruits add to promise for the future. From a parcel close to the series of Grand Cru vineyards, the wine shows the density of this fine vintage and its potential. Drink from 2024.

Figure 3: Samples from the wines recommendation task. Each item is associated with a review written by a professional wine reviewer. These two wines were also annotated as similar by a sommelier.

5.4 Image synthesis

We compare the performance of MTAdam with other optimizers, evaluated on three methods, pix2pix (Isola et al., 2017), CycleGan (Zhu et al., 2017) and SRGAN (Ledig et al., 2017a).

We used the learning rate as found in each method. Performance is evaluated using various metrics: L1, L2, PSNR, NMSE (normalized MSE), FID (Heusel et al., 2017), and SSIM (Wang et al., 2004). In each case, we follow the metrics used in the original work, with the addition of FID.

Pix2pix Experiments The objective function of the pix2pix generator has dual-terms:

$$\mathcal{L}_{pix2pix} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_{GAN} \quad (2)$$

	Facade images				maps→aerial			
	L1	**L2**	**NMSE**	**FID**	**L1**	**L2**	**NMSE**	**FID**
SGD-Momentum (equal weights)	120.12	18308	1.132	339.40	45.1	3428.3	0.491	102.45
RMSProp (equal weights)	53.98	4913	0.303	136.10	42.4	3056.7	0.435	95.74
Adam (equal weights)	54.92	5074	0.313	133.55	38.1	2632.3	0.368	88.08
MTAdam (equal weights)	46.50	3822	0.236	**130.38**	**33.6**	**2096.9**	**0.293**	86.40
Hyperband-FID	54.12	4992	0.308	141.96	35.14	2391.2	0.341	86.45
Adam-prescribed	**45.84**	**3811**	**0.235**	130.50	34.4	2163.4	0.303	**85.76**

Table 4: Comparing various methods for training pix2pix. Lower is better for all metrics.

		L1-A	**L2-A**	**NMSE-A**	**FID-A**	**L1-B**	**L2-B**	**NMSE-B**	**FID-B**
horse→zebra	SGD-Momentum-suboptimal-weights	82.3	10217	0.5866	283.3	127.0	20331	0.9898	406.3
	RMSProp-suboptimal-weights	73.8	8384	0.4814	233.9	70.6	7683	0.3740	230.9
	Adam-suboptimal-weights	74.4	8421	0.4935	229.6	72.3	7742	0.3842	227.1
	MTAdam-suboptimal-weights	68.7	**7235**	**0.4151**	73.0	69.0	7281	0.3545	**139.7**
	Hyperband-FID	72.3	8049	0.4687	189.2	71.8	7584	0.3746	192.6
	Adam-prescribed	**68.6**	7248	0.4161	**70.6**	**68.6**	**7207**	**0.3509**	143.8
maps→aerial	SGD-Momentum-suboptimal-weights	26.7	1102	0.0209	437.5	73.01	7146	1.0	738.6
	RMSProp-suboptimal-weights	12.9	389	0.0074	170.2	36.81	2260	0.3154	231.3
	Adam-suboptimal-weights	12.9	386	0.0073	166.3	37.00	2270	0.3161	234.2
	MTAdam-suboptimal-weights	10.6	376	0.0071	**113.0**	**35.37**	**2242**	**0.3137**	48.8
	Hyperband-FID	12.9	387	0.0073	169.4	36.91	2226	0.3102	233.1
	Adam-prescribed	**8.5**	**301**	**0.0057**	116.1	35.45	2250	0.3148	**46.8**

Table 5: Performance of the various optimization methods, when applied to the CycleGAN algorithm.

		Set14			BSD100		
		PNSR↑	**SSIM↑**	**FID↓**	**PSNR↑**	**SSIM↑**	**FID↓**
equal weights	SGD-Mmntum	22.95	0.6213	74.92	23.01	0.5997	79.45
	RMSProp	23.04	0.6252	72.92	23.14	0.6020	76.64
	Adam	23.43	0.6264	69.96	23.71	0.6070	77.24
	MTAdam	25.97	0.7219	**66.03**	**25.23**	**0.6690**	74.97
	Hyperband-FID	24.89	66.24	67.26	24.80	0.6554	75.65
	Adam-prescribed	**26.02**	**0.7397**	66.59	25.16	0.6688	**72.57**

Table 6: SRGAN results.

Where $\mathcal{L}_{GAN}$ is the GAN loss of the generator, $\mathcal{L}_1$ is the pixel loss, and λ_1 and λ_2 are set to 100 and 1, respectively. In our study, we train Pix2pix with equal weights, setting λ_1 to 100 (which implies a 1:1 ratio between the two terms).

Two datasets are used: matching facade images and their semantic labels (Tyleček and Šára, 2013) and aerial photographs and matching maps (Isola et al., 2017). Performance is reported on a holdout test set of each benchmark.

Fig. 4 depicts the test-performance of multiple models per epoch. The experiment's name contains 'equal-weights' for the case of equal loss terms, and 'prescribed' when using the prescribed λ values. As can be seen, MTAdam yields a similar convergence as the Adam-prescribed. Specifically, MTAdam converges substantially better than the Adam-equal-weights experiment, leading to improved L1 and FID scores.

In Tab. 4, MTAdam is compared with all baselines, applied for training pix2pix models. The top four models in each section utilize equal weighting, each applied with a different optimizer. The Hyperband experiment utilizes the FID metric to perform a hyperparameter search on λ_2.

The results of the table clearly show the advantage of MTAdam over all baseline methods. In addition, it also shows a slight improvement in performance in comparison to the usage of Adam on the prescribed weights. Appendix Fig. 8 presents two representative samples from the facades test

	MRPC	RTE	pix2pix facade				CycleGAN horse2zebra			
	Acc.	**Acc.**	**L1**	**L2**	**NMSE**	**FID**	**L1-A**	**FID-A**	**L1-B**	**FID-B**
(i) Removing L6–8	72.4	58.5	53.45	5021	0.326	134.0	74.7	225.2	71.2	217.1
(ii) Removing L6, changing g_ℓ^i to g^i	83.7	63.9	46.8	3850	0.258	133.5	70.1	75.2	69.5	145.7
(iii) No scaling by $n_{\ell,t}^1$ in L8	78.9	64.2	46.7	3862	0.242	136.3	72.3	82.4	70.6	167.2
(iv) Line 14: scaling like Adam	84.2	63.8	48.2	3956	0.245	171.5	73.5	210.5	71.1	190.3
(v) Line 14: scaling by mean instead of max	86.3	69.2	48.3	4020	0.249	156.8	73.7	208.4	71.6	173.5
Full method	**87.2**	**73.7**	**46.5**	**3822**	**0.236**	**130.4**	**68.7**	**73.0**	**69.0**	**139.7**

Table 7: Ablation study results.

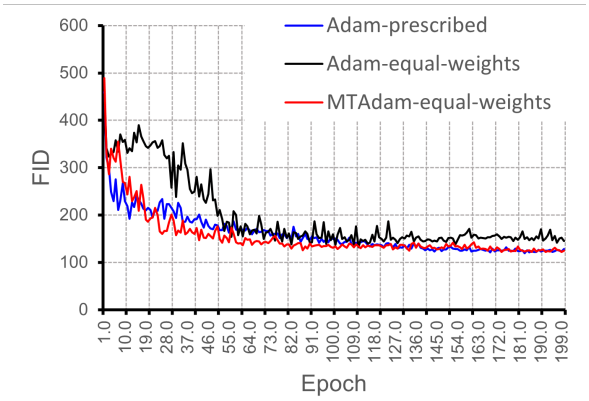


Figure 4: FID per epoch on the validation set of Facade for three pix2pix variants. Adam-prescribed and Adam-equal-weights employ the Adam optimizer with a loss combination that utilizes the prescribed weights and equal weights, respectively. MTAdam-equal-weights utilizes MTAdam with equal weights between losses. See appendix for the analog L1 plot.

set. Adam-equal-weights introduces visual artifacts and suffers from mode collapse (it generates the same corrupted patch in the top right corner of many images). MTAdam-equal-weights yields higher-quality images, similar to those of the original Pix2Pix, using the prescribed weights.

CycleGAN experiments The CycleGAN objective function is composed of six loss terms:

$$\mathcal{L}_{CycleGAN} = \lambda_1\mathcal{L}_{GAN\_A} + \lambda_1\mathcal{L}_{GAN\_B} + \lambda_2\mathcal{L}_{cycle\_A} + \lambda_2\mathcal{L}_{cycle\_B} + \lambda_3\mathcal{L}_{idt\_A} + \lambda_3\mathcal{L}_{idt\_B} \quad (3)$$

Where the $\mathcal{L}_{GAN}$, $\mathcal{L}_{cycle}$, $\mathcal{L}_{idt}$ terms are the GAN loss, cycle GAN loss and identity loss, for each one of the sides (A or B). λ_1, λ_2 and λ_3 are set to 1, 10 and 0.5, respectively. In our experiments, we employ CycleGAN with suboptimal weighting, by setting λ_2 to 1000, leaving λ_1 and λ_3 unchanged.

Appendix Fig. 4 compares the convergence of three CycleGAN models: (1-2) MTAdam and Adam, both with suboptimal weighting, and (3) Adam with the prescribed weights. All models are applied to the horse2zebra dataset (Deng et al., 2009). As can be seen, MTAdam exhibits a competitive convergence to the Adam experiment applied with the prescribed weights, which is much better than the performance of Adam with the suboptimal weights. Tab. 5 presents the performance of MTAdam applied on CycleGAN, compared to all five baselines, and evaluated on two datasets. MTAdam with the suboptimal weighting yields competitive performance to the Adam method, which uses the prescribed hyperparams.

Appendix Fig. 9 exhibits representative images from the CycleGAN models, showing that MTAdam, even when applied to a loss with suboptimal weights, matches or improves the results of Adam on the prescribed weights.

SRGAN The Super Resolution GAN (SRGAN) (Ledig et al., 2017a) objective function is:

$$\mathcal{L}_{SRGAN} = \lambda_1\mathcal{L}_{MSE} + \lambda_2\mathcal{L}_{GAN} + \lambda_3\mathcal{L}_{Perceptual} \quad (4)$$

for which the total loss is a combination of a GAN loss, perceptual loss and MSE. The λ_1, λ_2 and λ_3 are set to 1, 0.001 and 0.006. We employ a suboptimal combination between the loss terms, by setting $\lambda_2 = \lambda_3 = 1$.

We evaluate SRGAN trained with MTAdam and equal weights on two test sets, Set14 (Zeyde et al., 2010) and BSD100 (Martin et al., 2001). The results, listed in Tab. 6 demonstrate that MTAdam can effectively recover from suboptimal weights, while the other optimization methods suffer a degradation in performance.

5.5 Ablation Study

Tab. 7 presents an ablation study for suboptimal pix2pix on the facade images and for suboptimal

CycleGAN on the zebra2horse dataset. The following variants are considered (the descriptions refer to Alg. 1): (i) treating all layers as one layer and eliminating lines 6-8 altogether. (ii) training all layers at once, and performing the normalization in line 8 once for the entire gradient, i.e., still normalizing by the magnitude of the gradient of the first term. (iii) scaling the gradients of each layer l and each term i in line 8 by $(n^i_{l,t})^{-1}$ but not by $n^1_{l,t}$. (iv) replacing the term $\max(\widehat{v^1_t}...\widehat{v^I_t})$ in line 14 with $\widehat{v^i_t}$, in an analogous way to line 14 in Adam. (v) replacing the same term with the mean $I^{-1}\sum \widehat{v^i_t}$.

The results, shown in Tab. 7, indicate that it is crucial to employ a per layer analysis, in the way it is done in MTAdam, that normalizing by the magnitude of the gradient of an anchor term is highly beneficial, and that the maximal variance is a better alternative to other scaling terms.

6 Conclusions

MTAdam is shown to be a widely applicable optimizer, which can dynamically balance multiple loss terms in an effective way. It is a general algorithm, which can find its usage in additional types of tasks that require the optimization of multiple terms, such as domain adaptation and some forms of self-supervised learning. Our code is attached as supplementary. MTAdam is implemented as a generic pytorch optimizer and applying it is almost as simple as applying Adam.

Acknowledgment

This project has received funding from the European Research Council (ERC) under the European Union Horizon 2020 research and innovation programme (grant ERC CoG 725974). The contribution of the first author is part of a PhD thesis research conducted at Tel Aviv University.

References

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. 2014. Using meta-learning to initialize bayesian optimization of hyperparameters. In *Proceedings of the 2014 International Conference on Meta-learning and Algorithm Selection-Volume 1201*, pages 3–10. Citeseer.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017a. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017b. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Itzik Malkiel, Oren Barkan, Avi Caciularu, Noam Razin, Ori Katz, and Noam Koenigstein. 2020a. Optimizing bert for unlabeled text-based items similarity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1704–1714.

Itzik Malkiel, Oren Barkan, Avi Caciularu, Noam Razin, Ori Katz, and Noam Koenigstein. 2020b. Recobert: A catalog language model for text-based recommendations. *arXiv preprint arXiv:2009.13292*.

David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. 2017. The numerics of gans. In *Advances in Neural Information Processing Systems*, pages 1825–1835.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. 2009. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Radim Tyleček and Radim Šára. 2013. Spatial pattern templates for recognition of objects with regular structure. In *German Conference on Pattern Recognition*, pages 364–374. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

Zi Wang, Bolei Zhou, and Stefanie Jegelka. 2016. Optimization as estimation with gaussian processes in bandit settings. In *Artificial Intelligence and Statistics*, pages 1022–1031.

Roman Zeyde, Michael Elad, and Matan Protter. 2010. On single image scale-up using sparse-representations. In *International conference on curves and surfaces*, pages 711–730. Springer.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

Supplementary Appendices[3]

A Motivation

Following Sec. 3 in the main text, Fig.5 presents the gradients' magnitude of the two learned parameters of the polynomial $f(x) = ax + bx^2$ w.r.t. the loss function. Fig.6 presents the FID scores vs. epoch, for the different SRGAN experiments described in the same section.

B Image synthesis

Fig. 7, depicts the test-performance of multiple pix2pix models per epoch. The experiment's name contains 'equal-weights' for the case of equal loss terms, and 'prescribed' when using the prescribed λ values. As can be seen, MTAdam yields a similar convergence as the Adam-prescribed, measured by the L1 scores, and converges substantially better than the Adam-equal-weights (we observe a similar trend in the FID, as presented in the main text).

The pix2pix Hyperband experiments incorporate 40 trials (i.e. 40 training processes, which gives it a great advantage), each trial randomly sampled a different lambda from the range $[10^{-4}, 10^{4}]$. We report Hyperband performance, by utilizing the trial that is associated with the chosen lambda. In the aerial experiment, the Hyperband failed to choose a λ_2 value that is close to the original value of 1. In the facade experiment, Hyperband sampled at least one lambda value between 0.5 to 10, yet the retrieved best model utilizes a higher lambda value of 162.89, since this value showed a preferable FID

[3] Put here for the reader's convenience.

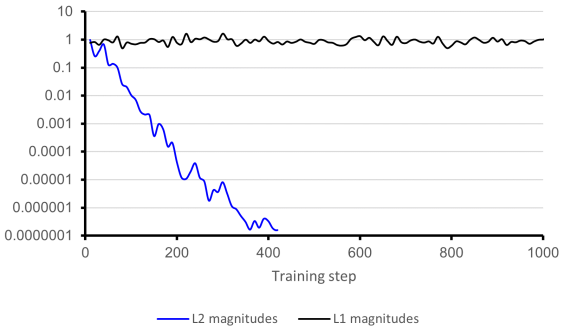


Figure 5: Training dynamics in a two neurons network, learning the coefficients of a polynomial of the form $f(x) = ax + bx^2$, minimizing L1, L2 or L1 where the gradients are scaled with the magnitudes of the L2 gradients. The plot presents the gradients' magnitude of the two parameters w.r.t. the loss function (i.e. $\|(\frac{\partial L1}{\partial W_1}, \frac{\partial L1}{\partial W_2})\|_2$ and $\|(\frac{\partial L2}{\partial W_1}, \frac{\partial L2}{\partial W_2})\|_2$ for the L1 and L2 experiments, respectively).

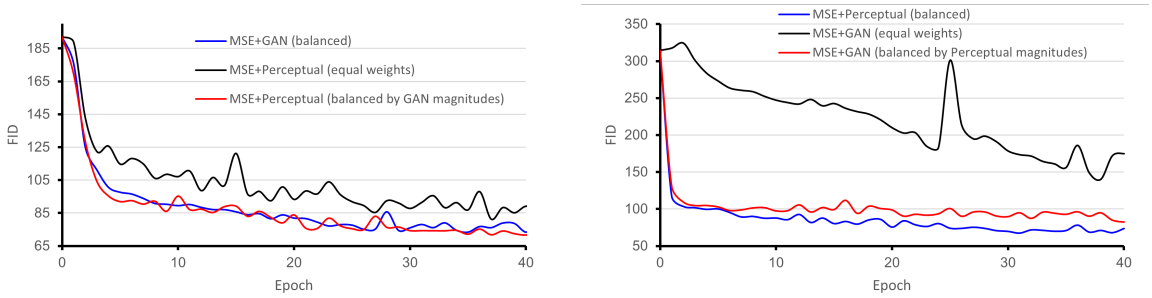
score on the validation set. In the maps dataset, a value of 59.61 was selected.

Fig. 8 compares the convergence of three CycleGAN models: unbalanced MTAdam, unbalanced Adam, and balanced Adam. All models are applied on the horse2zebra dataset (Deng et al., 2009). As can be seen, MTAdam exhibits a competitive convergence to the Adam experiment applied with balanced weighting, which is much better than the performance of Adam on the unbalanced weights.

Fig. 9 presents a few representative samples from the CycleGAN(Zhu et al., 2017) experiments, employed to transfer horse images to zebras. As can be seen, the unbalanced Adam training completely fails to generate zebras images and collapses to the identity mapping. This can be attributed to the domination of cyclic loss in the unbalanced settings, which dictates the convergence, leaveing the other loss terms ineffective. On the other hand, the unbalanced MTAdam model was able to successfully generate zebra images, in a quality that is similar or better to the balanced adam model (which uses the prescribed weights). In particular, in the first row, we can see that CycleGAN-Unbalanced-MTAdam was able to outperform the CycleGAN-Balanced-Adam experiment, as the latter fails to generate a zebra image for this particular sample, while MTAdam was able to generate a fairly good quality image.

Fig. 10 depicts a few samples from the same models described above, this time employed to generate horse images from zebra images. As can be seen, the unbalanced Adam fails again to generate horse images, collapses to the identity mapping, and this time also introduces visual artifacts in a few images (see the green artifact in the bottom row). In addition, MTAdam yields images of the same quality as the balanced Adam experiment.

Fig. 11 presents visual results for the SRGAN (Ledig et al., 2017a) experiments. The images were taken from Set14 (Zeyde et al., 2010). All models were trained to generate high-resolution images from low-resolution images, with a factor of 4x upscaling. As can be seen, the SRGAN model that employs unbalanced weights and Adam optimizer yields images with visual artifacts, and low fidelity. This can be attributed to the domination of the GAN and perceptual loss terms over the pixel-wise term. In contrast, the SRGAN model that employs our MTAdam optimizer with unbalanced weights yields images of the same quality as the SRGAN that utilizes the prescribed weights (Ledig et al., 2017a) and Adam.



(a) (b)

Figure 6: We train a superresolution image mapping technique using two out of three loss terms: MSE, GAN, and perceptual loss. (a) the GAN loss is well balanced and we obtain effective training when combining it with the MSE loss. However, the weight of the perceptual loss was naively set to 1, which is about two orders of magnitude larger than the optimal value for combining it with the MSE loss. This leads to poor SSIM scores (shown in the main text) and FID scores (shown in this figure). However, when scaling the perceptual loss gradients by the magnitude of the gradients of the GAN loss, the perceptual loss supports the MSE loss. (b) same, where the roles of the perceptual loss and the GAN loss are reversed.

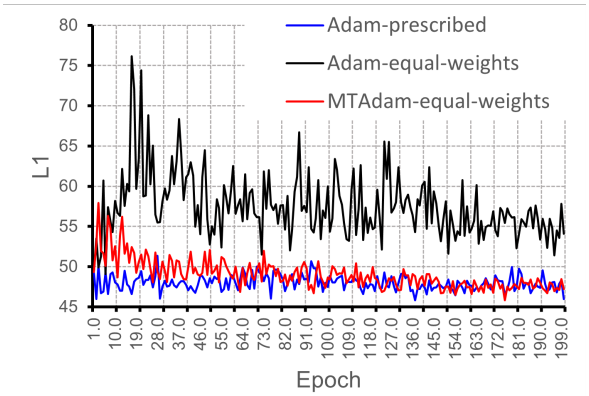


Figure 7: L1 per epoch on the validation set of Facade for three pix2pix variants. Adam-prescribed and Adam-equal-weights employ the Adam optimizer with a loss combination that utilizes the prescribed weights and equal weights, respectively. MTAdam-equal-weights utilizes MTAdam with equal weights between losses. See the main text for the analog FID plot.

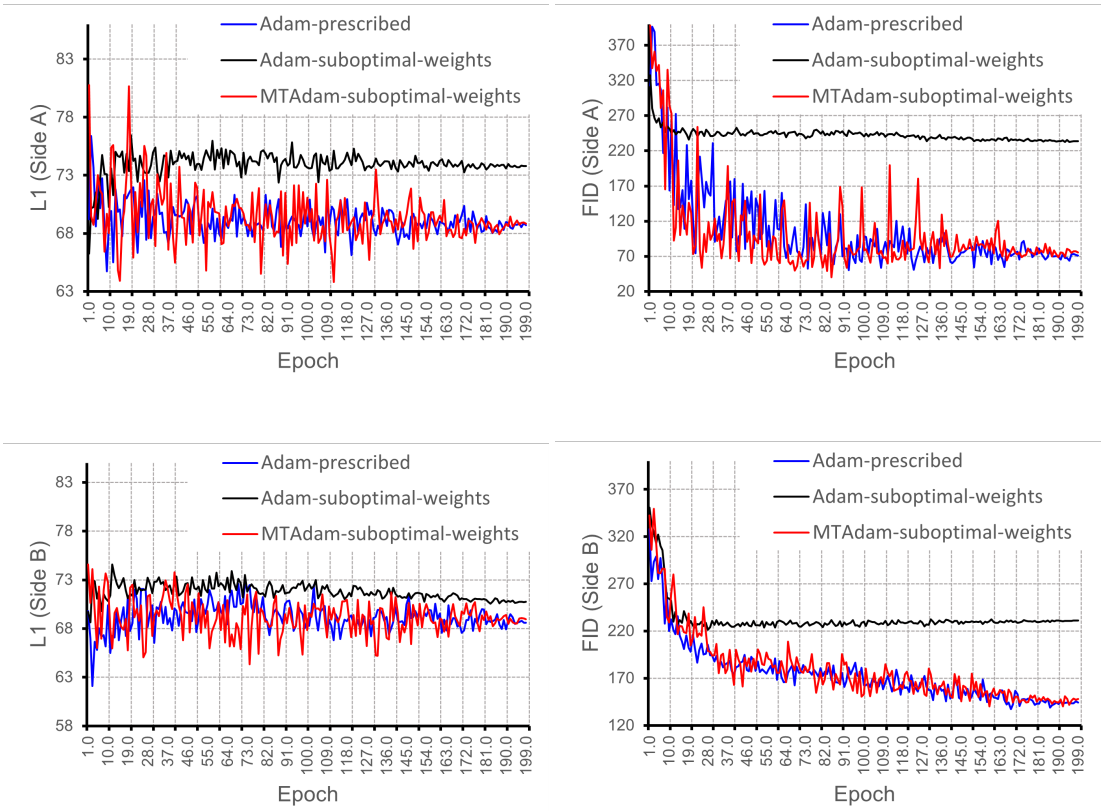


Figure 8: The L1 (left) and FID (right) metrics evaluated per epoch CycleGAN models trained on the horse2zebra dataset (Deng et al., 2009). (top) generated zebra images. (bottom) generated horse images.

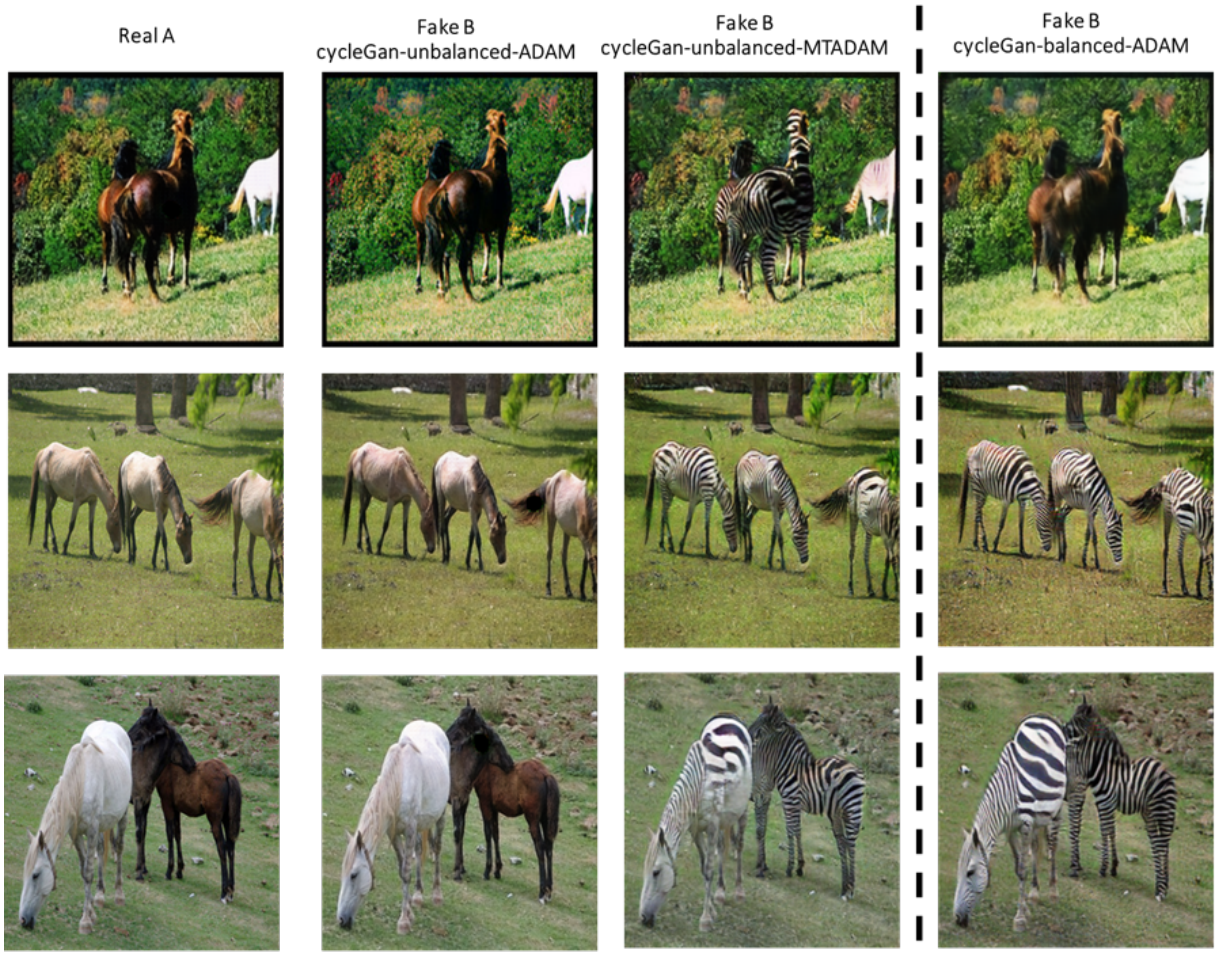


Figure 9: Samples from the CycleGAN experiments employed on the zebra2horse dataset, mapping horse images to zebras. CycleGAN-Balanced-Adam employs CycleGAN with the prescribed weights (Zhu et al., 2017). CycleGAN-Unbalanced-Adam runs with unbalanced initial weights, which fails to generate zebra images. CycleGAN-Unbalanced-MTAdam (our method) starts with unbalanced weights but produces images of the same quality as CycleGAN-Balanced-Adam.

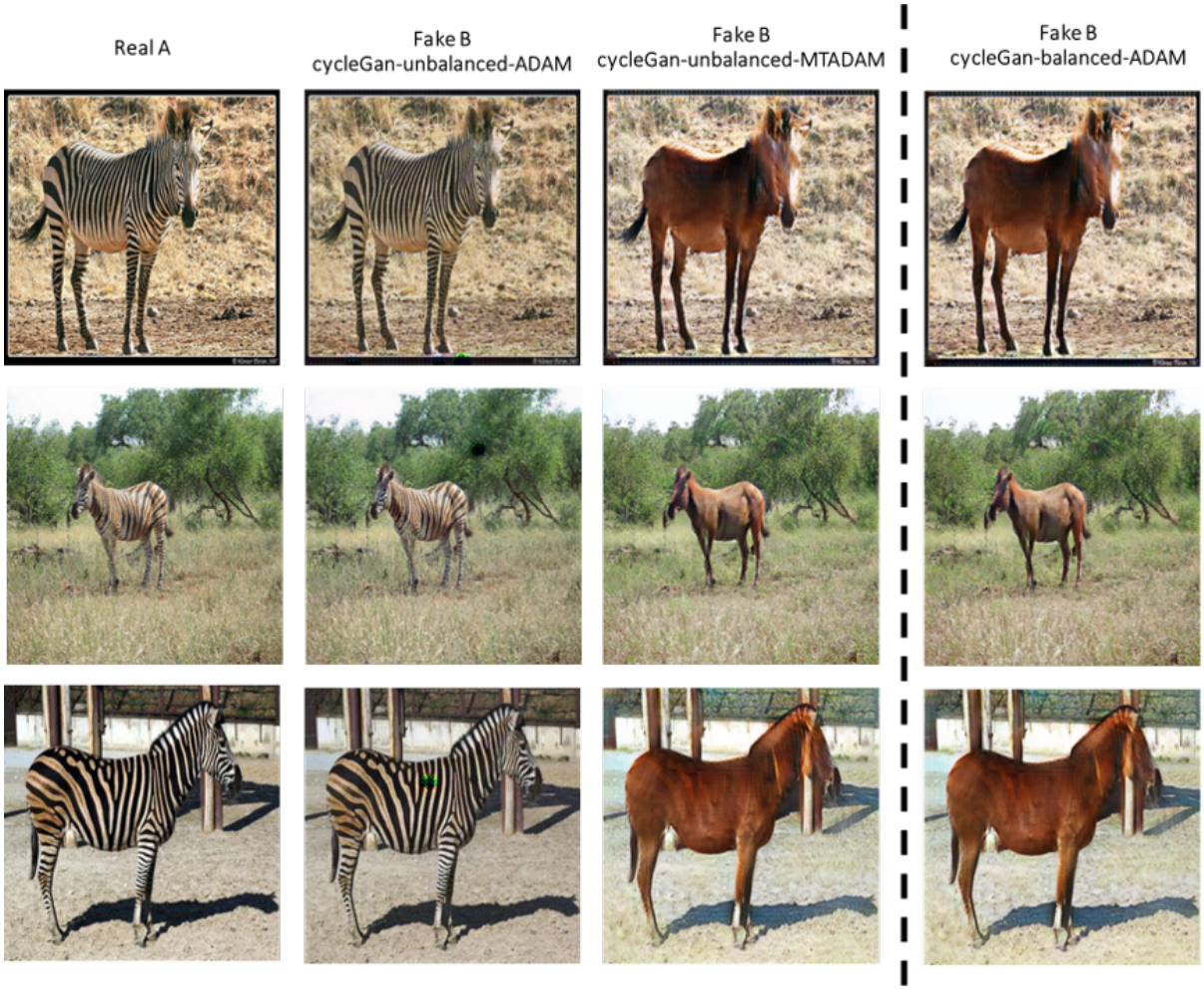


Figure 10: Samples from the CycleGAN experiments employed on the zebra2horse dataset, mapping zebra images to horses. CycleGAN-Balanced-Adam employs CycleGAN with the prescribed weights (Zhu et al., 2017). CycleGAN-Unbalanced-Adam runs with unbalanced initial weights, fails to generate horse images, collapses to identity mapping, and introduces visual artifacts. CycleGAN-Unbalanced-MTAdam (our method) starts with unbalanced weights but produces images of the same quality as CycleGAN-Balanced-Adam.

Figure 11: Samples from the test set of the SRGAN(Ledig et al., 2017a) experiments. Images were taken from Set14(Zeyde et al., 2010). SRGAN-Unbalaced-Adam employs SRGAN training with the unbalanced initial weights, as described in the main text. SRGAN-Balanced-Adam employs SRGAN with the prescribed weights. SRGAN-Unbalanced-MTAdam utilizes our proposed MTAdam optimizer, runs with unbalanced initial weights but produces images of the same quality of SRGAN-Balanced-Adam.

Figure 12: Samples from the Facade test set. Adam-prescribed employs pix2pix with the prescribed weights (Isola et al., 2017). Adam-equal-weights runs with equal initial weights, which leads to visual artifacts and low fidelity (see the top right patch). MTAdam-equal-weights (our method) starts with equal weighting but produces images of the same quality as Adam-prescribed.

(a) (b) (c) (d) (e)

Figure 13: CycleGAN results for mapping (a) real maps to fake images in the aerial photos domain for (b) Adam with suboptimal weights, (c) MTAdam with suboptimal weights, and (d) Adam with the prescribed weights. (e) the ground truth.