REVISE SATURATED ACTIVATION FUNCTIONS

Bing Xu, Ruitong Huang
Department of Computing Science
University of Alberta
{antinucleon,rtonghuang}@gmail.com

Mu Li
Computer Science Department
Carnegie Mellon University
muli@cs.cmu.edu

ABSTRACT

In this paper, we revise two commonly used saturated functions, the logistic sigmoid and the hyperbolic tangent (tanh). We point out that, besides the well-known non-zero centered property, slope of the activation function near the origin is another possible reason making training deep networks with the logistic function difficult to train. We demonstrate that, with proper rescaling, the logistic sigmoid achieves comparable results with tanh. Then following the same argument, we improve tahn by penalizing in the negative part. We show that "penalized tanh" is comparable and even outperforms the state-of-the-art non-saturated functions including ReLU and leaky ReLU on deep convolution neural networks.

Our results contradict to the conclusion of previous works that the saturation property causes the slow convergence. It suggests further investigation is necessary to better understand activation functions in deep architectures.

1 INTRODUCTION

Activation functions play an important role in artificial neural networks in that they help bring non-linearity to the networks. Different activation functions can significantly affect the performance of a neural network, and therefore how to choose a good activation function has attracted lots of studies in the literature. One of the publicly accepted arguments is that a saturated activation function may cause the gradient vanishing (and/or explosion), and thus is less preferred. In particular, it has been reported that the backpropagated gradient of a network which uses the logistic sigmoid $f(x) = 1/(1 + e^{-x})$ as its activation function may vanish or explode quickly. Currently there is no success, to our best knowledge, in training a deep neural network with this activation function without Batch Normalization(Ioffe & Szegedy, 2015). However, different from logistic sigmoid, the performances of networks with tanh, which is also saturated, are much more stable. For example, a deep convolution neural network using tanh is able to reach a local optimality with careful Layer-sequential unit-variance weight initialization (Mishkin & Matas, 2015). It then raise a question what makes these two functions so different, despite that they are both saturated by which our investigations in the paper are driven.

We start with verifying the assumptions that are required in Xavier initialization (Glorot & Bengio, 2010) for a general activation function in its linear regime, based on which we discuss the failure of logistic sigmoid and propose two methods to overcome the training problem of the deep Sigmoid networks. Our analysis suggests that besides the well-known non-zero centered property, slope and the offset of the activation function near the origin is another possible reason causing the vanishing (and/or explosion) of the gradient.

One well accepted explanation about the empirical success of ReLU is about its non-saturated property, compared to other saturated functions. Similarly, we design new activation functions to investigate the essential effects of being leaky, its linear regime, and its saturation property outside the linear regime of an activation function. In particular, we compare the performance of a new activation function, called leaky tanh, to the performances of ReLU and leaky ReLU. The new function penalized tanh shares similar property in its linear regime with leaky ReLU, yet different from ReLU or leaky ReLU, it is saturated outside its linear regime. Our results provide more insights about the effect of different activation functions on the performance of the neural networks, and suggest that further investigation is still needed for better understanding.

All the networks in the paper are trained by using MXNet (Chen et al., 2015).

2 WHY TRAINING DEEP NEURAL NETWORKS IS HARD WITH THE LOGISTIC SIGMOID

We first investigate the behaviors of the activation variance and the gradient variance, for a general activation function in its linear regime, within the theoretical framework developed in (Glorot & Bengio, 2010). Our analysis is focusing on the initialization stage. Basis on the analysis, conditions that are required for the activation function to maintain the activation variance and the gradient variance are presented, which then leads to the discussion about the difficulty of training a deep neural networks using the logistic sigmoid.

To simplify the analysis, we assume the following fully connected neural network. Let $y^{(l)} \in \mathbb{R}^{n_l}$ be the output of layer l for $l = 1, 2, \ldots, L$, and f be the activation function, in a forward pass we have

$$x^{(l)} = W^{(l-1)}y^{(l-1)} + b^{(l-1)} \tag{1}$$

$$y^{(l)} = f(x^{(l)}) \tag{2}$$

$$\tag{3}$$

where the weight $W^{(l-1)} \in \mathbb{R}^{n_l \times n_{l-1}}$ and the bias $b^{(l-1)} \in \mathbb{R}^{n_l}$.

Assume that we initialized $b^{(l-1)}$ to be 0. Thus the randomness of $y^{(l)}$ comes from both the output of previous layer $y^{(l-1)}$ and the weight $W^{(l-1)}$ which is randomly initialized. Further assume that for each $W^{(l)}$, its elements are initialized independently with $\mathbb{E}\left[W^{(l)}\right] = 0$ with equal variance σ_l^2. Also assume that all the weights $\{W^{(l)}; l = 0, \ldots, L-1\}$ are mutually independent.

Proposition 1. *Assume that $\frac{\partial cost}{\partial y^{(L)}}$ is independent to $W^{(t)}$, $t = 1, \ldots, L-1$, with Var $\left(\frac{\partial cost}{\partial y^{(L)}}\right) = d_L I$ for some constant d_L. Also assume that $\mathbb{E}\left[\frac{\partial cost}{\partial y^{(L)}}\right] = 0$. Let the activation function $f(x) = \alpha x + \beta$, then for any layer l,*

$$Var\left(y^{(l)}\right) = \alpha^2 n_{l-1}\sigma_{l-1}^2\left(Var\left(y^{(l-1)}\right) + \beta^2 I_{n_l}\right). \tag{4}$$

$$Var\left(\frac{\partial cost}{\partial y^{(l-1)}}\right) = \alpha^2 n_l \sigma_{l-1}^2 Var\left(\frac{\partial cost}{\partial y^{(l)}}\right). \tag{5}$$

The proof is formally developed in Appendix 5, following the idea of Glorot & Bengio (2010). Assume that the network is initialized using Xavier initialization, then to maintain the activation variance and the gradient variance, namely for $l = 1, \ldots, L$,

$$\text{Var}\left(y^{(l)}\right) = \text{Var}\left(y^{(l-1)}\right) \quad \text{and} \quad \text{Var}\left(\frac{\partial \text{cost}}{\partial y^{(l)}}\right) = \text{Var}\left(\frac{\partial \text{cost}}{\partial y^{(l-1)}}\right);$$

$$n_l \sigma_{l-1}^2 \cong 1 \quad \text{and} \quad n_{l-1}\sigma_{l-1}^2 \cong 1;$$

α and β must satisfy that

$$\alpha = 1 \quad \text{and} \quad \beta = 0.$$

Now consider the Taylor expansions of different activation functions,

$$\text{sigmoid}(x) = \frac{1}{2} + \frac{x}{4} - \frac{x^3}{48} + O(x^5) \tag{6}$$

$$\tanh(x) = 0 + x - \frac{x^3}{3} + O(x^5) \tag{7}$$

$$\text{relu}(x) = 0 + x \quad \text{for } x \geq 0. \tag{8}$$

It shows that in their linear regimes, both tanh and relu have the desirable property that $\alpha = 1$ and $\beta = 0$. If the weight is initialized with zero mean and $1/n$ variance, which is one of the widely used method Glorot & Bengio (2010), then both the forward output and backward gradient will be

in proper range at least for the first few iterations. However, this is not true for logistic sigmoid. First, its slope in the linear regime is $1/4$ rather than 1, then we need to initialize the weight is a 16 times smaller variance to keep the each layer's gradient variance the same. Second, it has a non-zero mean, which makes the output variance increase linear with the layer.

One simply way to fix this problem is rescale the logistic sigmoid to match the first two degree coefficient with both tanh and relu, so the weight initialization method used for the latter two can be applied to the logistic sigmoid. In other word, we transform the logistic sigmoid by

$$\text{scaled sigmoid}(x) = 4 \times \text{sigmoid}(x) - 2 = \frac{4}{1 + e^{-x}} - 2 \tag{9}$$

The scaled logisitc sigmoid is illustrated in Figure 1. As can bee seen, it is similar to tanh near 0, but the saturation value is two times larger than tanh. Our experimental results shows that the scaled sigmoid function achievess comparable results with tanh.

In the other perspective, the scale factor 4 in scaled sigmoid function is equivalent to scale initialization and learning rate by factor of 4; the bias term -2 in scaled sigmoid function is equivalent to fixed bias after linear transform.
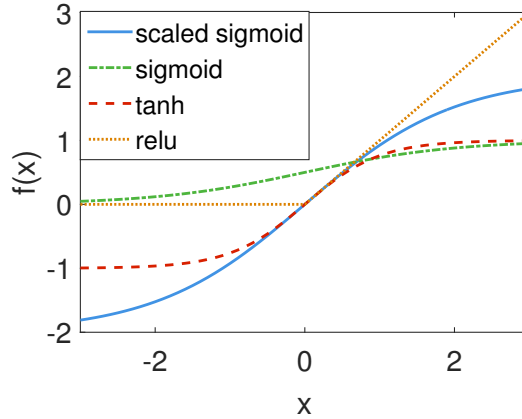

Figure 1: Compare the scaled logistic sigmoid with other activation functions.

3 PENALIZED SATURATED ACTIVATION FUNCTIONS

Recently, several variants have been proposed in the literature to improve the performance of non-saturated activation functions. One that we are particularly interested in is the 'leaky ReLU', which is as follows.

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases} \tag{10}$$

where $a \in (0, 1)$. Comparing to the standard ReLU, leaky ReLU gives nonzero gradient for negative value. Although its output is no longer sparse, which is claimed to be the main advantage of ReLU, recent works show that oftentimes leaky ReLU outperforms ReLU (Xu et al., 2015; Clevert et al., 2015).

On the other viewpoint, the leaky ReLU can be viewed as an improvement over the simple identical activation function $f(x) = x$ which penalizes the gradient of the negative part. Inspired by this observation, we propose to also penalize the negative part of the saturated activation functions. In particular, we propose "penalized tanh" which takes the form

$$f(x) = \begin{cases} \tanh(x) & \text{if } x > 0 \\ a \cdot \tanh(x) & \text{otherwise} \end{cases} \tag{11}$$

3

where $a \in (0, 1)$. Figure 2 compares the penalized tanh with other activation functions. As can be seen, if the same a is used, the penalized tanh can be viewed as a saturated version of leaky ReLU. These two functions have similar value near 0, since both function share the same Taylor expansion up to the first order. But different to leaky Relu, penalized tanh saturates to $-a$ and 1 when moving away from 0.
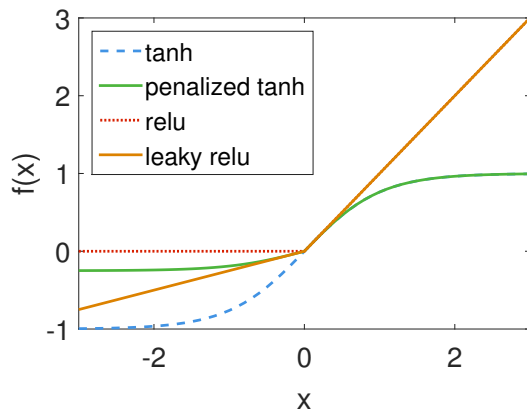

Figure 2: Compare the penalized tanh with other functions. Both penalized tanh and leaky ReLU uses $a = 1/4$.

4 EXPERIMENT

Activation	Train Accuracy	Test Accuracy
sigmoid	diverged	diverged
scaled sigmoid	89.39%	59.11%
tanh	96.94%	61.99%
ReLU	99.17%	67.91%
penalized tanh (a = 0.25)	99.75%	70.43%
leaky ReLU (a = 0.25)	99.85%	70.64%

Table 1: Vary activation function for inception network on CIFAR-100
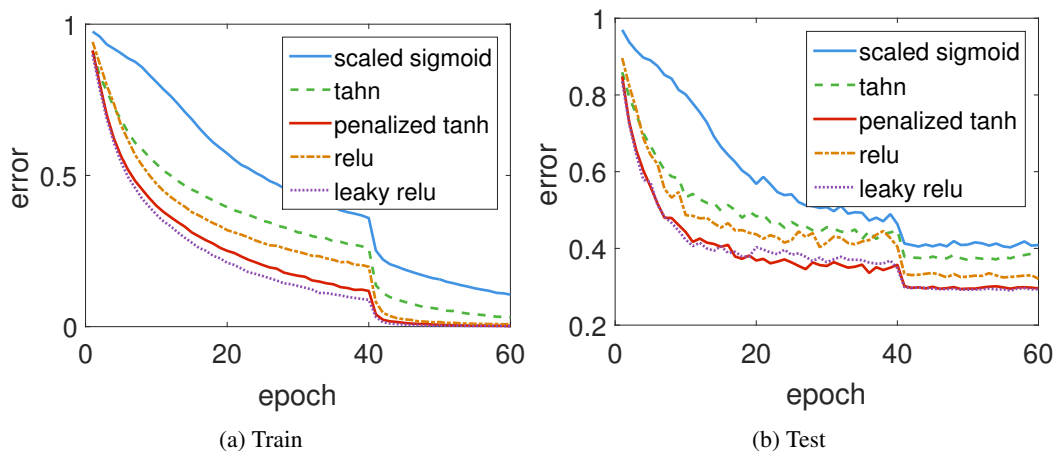

(a) Train

(b) Test

Figure 3: Error versus epoch for inception network with various activation function on CIFAR-100.

We evaluated the proposed two activation functions on a 33-layers inception network without batch normalization (Ioffe & Szegedy, 2015). We used the CIFAR-100 dataset, which is an image classification dataset with 100 classes. All networks are initalized with (Glorot & Bengio, 2010).

The converge results are shown in Figure 3 and the final training and test accuracy is reported in Table 1. As expected, ReLU converges faster than tanh but is outperformed by leaky ReLU. For the proposed functions, the scaled logistic sigmoid successfully converges into a local minimal. Its performance is close its cousin tanh, both of them are saturated and have have similar shape near 0. On the other hand, the penalized tanh converges more than 2 times faster than the standard tanh. It gives almost identical results with its non-saturated version, namely leaky ReLU.

Based on the observations, it seems that the performance of various activation functions on the inception network mainly depend on the function shape near 0, namely the values of $f(0)$ and $\partial f(0)$. It is possibly due to the inputs to the activation function are near 0.

5 CONCLUSION & FUTURE WORK

The result of this paper is two-fold. We first attempt to explain and fix the failure of training a deep Sigmoid network, based on the idea of the work (Glorot & Bengio, 2010). A re-scaled Sigmoid activation is proposed in the paper to make deep Sigmoid network trainable. The other result of this paper is to investigate the differences in network performances between using saturated activation function and using non-saturated ones. Our result suggests that when using penalization on negative part, saturation of the activation function is comparable to ReLU and Leaky ReLU. There are still many open questions requiring further investigation: 1. How to efficiently determine different learning rates for different layers in a very deep neural network? 2. How does the positive part (on $[0, +\infty)$) and the negative part (on $(-\infty, 0]$) of the activation function affect the performance of the network?

REFERENCES

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

APPENDIX

Proof of Proposition 1. First note that for each $l = 1 \ldots, L$,

$$\mathbb{E}\left[x^{(l)}\right] = \mathbb{E}\left[W^{(l-1)}y^{(l-1)} + b^{(l-1)}\right] = \mathbb{E}\left[W^{(l-1)}\right]\mathbb{E}\left[y^{(l-1)}\right] + 0 = 0,$$

where the last equality is due to $\mathbb{E}\left[W^{(l-1)}\right] = 0$. Therefore,

$$\mathbb{E}\left[y^{(l)}\right] = \mathbb{E}\left[f(x^{(l)})\right] = \mathbb{E}\left[\alpha x^{(l)} + \beta\mathbf{1}\right] = \alpha\mathbb{E}\left[x^{(l)}\right] + \beta\mathbf{1} = \beta\mathbf{1}$$

To compute the variance of $y^{(l)}$, by definition

$$\text{Var}\left(y^{(l)}\right) = \text{Var}\left(\alpha x^{(l)} + \beta \mathbf{1}\right) = \alpha^2 \text{Var}\left(x^{(l)}\right).$$

Note that $y^{(l-1)}$ only depends on the weights $W^0, \ldots, W^{(l-2)}$, thus is independent to $W^{(l-1)}$. plugging the definition of $x^{(l)}$,

$$\begin{aligned}
\text{Var}\left(x^{(l)}\right) &= \text{Var}\left(W^{(l-1)}y^{(l-1)}\right) \\
&= \mathbb{E}\left[W^{(l-1)}y^{(l-1)}y^{(l-1)\top}W^{(l-1)\top}\right] - \mathbb{E}\left[W^{(l-1)}y^{(l-1)}\right]\mathbb{E}\left[W^{(l-1)}y^{(l-1)}\right]^\top \\
&= \mathbb{E}\left[W^{(l-1)}y^{(l-1)}y^{(l-1)\top}W^{(l-1)\top}\right] - \mathbb{E}\left[W^{(l-1)}\right]\mathbb{E}\left[y^{(l-1)}\right]\mathbb{E}\left[y^{(l-1)}\right]^\top\mathbb{E}\left[W^{(l-1)}\right]^\top \\
&= \mathbb{E}\left[W^{(l-1)}y^{(l-1)}y^{(l-1)\top}W^{(l-1)\top}\right],
\end{aligned}$$

where the third equality is due to the independence of $y^{(l-1)}$ and $W^{(l-1)}$, and the last equality is due to $\mathbb{E}\left[W^{(l-1)}\right] = 0$.

We will prove that the covariance matrix of $y^{(l)}$ is $c_l I$ for some constant c_l, $l = 1, \ldots, L$, using mathematical induction. For the base case $y^{(0)}$ being the input, by assumption the claim holds. Now assume that the claim holds for $l - 1$, i.e. $\text{Var}\left(y^{(l-1)}\right) = c_{l-1}I$, we would like to prove that $\text{Var}\left(y^{(l)}\right) = c_l I$ for some constant c_l. To simplify the proof, we would use Lemma 1 as follows.

Lemma 1. *Given that $\text{Var}(y) = CI_n$ and y is independent to $W \in \mathbb{R}^{m \times n}$, if the elements of W are mutually independent with common variance σ^2, then*

$$\mathbb{E}\left[Wyy^\top W^\top\right] = (Cn\sigma^2 + \sigma^2\|\mathbb{E}\left[y\right]\|_2^2)I_m.$$

Thus,

$$\begin{aligned}
\text{Var}\left(x^{(l)}\right) &= \mathbb{E}\left[W^{(l-1)}y^{(l-1)}y^{(l-1)\top}W^{(l-1)\top}\right] \\
&= (c_{l-1}n_{l-1}\sigma_{l-1}^2 + \sigma_{l-1}^2\|\mathbb{E}\left[y\right]\|_2^2)I_{n_l} \\
&= n_{l-1}(c_{l-1} + \beta^2)\sigma_{l-1}^2 I_{n_l}.
\end{aligned}$$

Picking $c_l = n_{l-1}(c_{l-1} + \beta^2)\sigma_{l-1}^2$ and the claim holds. Therefore,

$$\text{Var}\left(y^{(l)}\right) = \alpha^2 n_{l-1}(c_{l-1} + \beta^2)\sigma_{l-1}^2 I_{n_l} = \alpha^2 n_{l-1}\sigma_{l-1}^2\left(\text{Var}\left(y^{(l-1)}\right) + \beta^2 I_{n_l}\right).$$

Next we consider the computation of $\text{Var}\left(\frac{\partial \text{cost}}{\partial y^{(l-1)}}\right)$. Note that

$$\frac{\partial \text{cost}}{\partial y^{(l-1)}} = \frac{\partial y^{(l)}}{\partial y^{(l-1)}}\frac{\partial \text{cost}}{\partial y^{(l)}}.$$

By definition,

$$\frac{\partial y^{(l)}}{\partial y^{(l-1)}} = \frac{\partial x^{(l)}}{\partial y^{(l-1)}}\frac{\partial y^{(l)}}{\partial x^{(l)}} = \alpha W^{(l-1)}$$

Thus, $\frac{\partial y^{(l)}}{\partial y^{(l-1)}}$ is independent to $\frac{\partial \text{cost}}{\partial y^{(l)}}$ which only depends on $W^{(t)}$ for $t = l, \ldots, L$. Therefore, given that $\mathbb{E}\left[\frac{\partial \text{cost}}{\partial y^{(L)}}\right] = 0$,

$$\begin{aligned}
\mathbb{E}\left[\frac{\partial \text{cost}}{\partial y^{(l-1)}}\right] &= \mathbb{E}\left[\alpha W^{(l-1)}\frac{\partial \text{cost}}{\partial y^{(l)}}\right] \\
&= \alpha\mathbb{E}\left[W^{(l-1)}\right]\mathbb{E}\left[\frac{\partial \text{cost}}{\partial y^{(l)}}\right] \\
&= 0.
\end{aligned}$$

We can now compute the variance of $\frac{\partial \text{cost}}{\partial y^{(l-1)}}$.

$$\text{Var}\left(\frac{\partial \text{cost}}{\partial y^{(l-1)}}\right) = \text{Var}\left(\alpha W^{(l-1)}\frac{\partial \text{cost}}{\partial y^{(l)}}\right)$$

$$= \alpha^2 \text{Var}\left(W^{(l-1)}\frac{\partial \text{cost}}{\partial y^{(l)}}\right)$$

$$= \alpha^2 \mathbb{E}\left[W^{(l-1)}\frac{\partial \text{cost}}{\partial y^{(l)}}\frac{\partial \text{cost}}{\partial y^{(l)}}^\top W^{(l-1)\top}\right]$$

Note that by assumptions, $\frac{\partial \text{cost}}{\partial y^{(L)}}$ is independent to $W^{(t)}$, $t = 1, \ldots, L-1$, with $\text{Var}\left(\frac{\partial \text{cost}}{\partial y^{(L)}}\right) = d_L I$ for some constant d_L. Also $\mathbb{E}\left[\frac{\partial \text{cost}}{\partial y^{(l)}}\right] = 0$. Similar to the first part of the proof, given that $\frac{\partial \text{cost}}{\partial y^{(L)}}$ is independent to $W^{(t)}$, $t = 1, \ldots, L-1$, with $\text{Var}\left(\frac{\partial \text{cost}}{\partial y^{(L)}}\right) = d_L I$ for some constant d_L, one can prove that $\text{Var}\left(\frac{\partial \text{cost}}{\partial y^{(l)}}\right) = d_l I$ for some constant d_l, and thus

$$\text{Var}\left(\frac{\partial \text{cost}}{\partial y^{(l-1)}}\right) = \alpha^2(d_l n_l + \|\mathbb{E}\left[\frac{\partial \text{cost}}{\partial y^{(l)}}\right]\|_2^2)\sigma_{l-1}^2 I_{n_{l-1}} = d_l n_l \sigma_{l-1}^2 I_{n_{l-1}} = \alpha^2 n_l \sigma_{l-1}^2 \text{Var}\left(\frac{\partial \text{cost}}{\partial y^{(l)}}\right).$$

$\square$

Proof of Lemma 1.

$$\mathbb{E}\left[Wyy^\top W^\top\right] = \mathbb{E}\left[Wyy^\top W^\top - W\mathbb{E}\left[y\right]\mathbb{E}\left[y\right]^\top W^\top\right] + \mathbb{E}\left[W\mathbb{E}\left[y\right]\mathbb{E}\left[y\right]^\top W^\top\right]$$

$$= \mathbb{E}\left[W\text{Var}\left(y\right)W^\top\right] + \mathbb{E}\left[W\mathbb{E}\left[y\right]\mathbb{E}\left[y\right]^\top W^\top\right]$$

$$= C\mathbb{E}\left[WW^\top\right] + \mathbb{E}\left[W\mathbb{E}\left[y\right]\mathbb{E}\left[y\right]^\top W^\top\right].$$

Here the last equality is due to $\text{Var}\left(y\right) = cI_n$. For the first term, consider its (i,j)th element

$$\mathbb{E}\left[W_{i:}W_{j:}^\top\right] = \begin{cases} 0 & \text{if } i \neq j \\ n\sigma^2 & \text{if } i = j \end{cases},$$

where $W_{i:}$ is the ith row of W. Similarly for the second term $\mathbb{E}\left[W\mathbb{E}\left[y\right]\mathbb{E}\left[y\right]^\top W^\top\right]$, its (i,j)th element

$$\mathbb{E}\left[W_{i:}\mathbb{E}\left[y\right]\mathbb{E}\left[y\right]^\top W_{j:}^\top\right] = \mathbb{E}\left[y\right]^\top \mathbb{E}\left[W_{i:}^\top W_{j:}\right]\mathbb{E}\left[y\right] = \begin{cases} 0 & \text{if } i \neq j \\ \sigma^2\|\mathbb{E}\left[y\right]\|_2^2 & \text{if } i = j \end{cases}.$$

Therefore,

$$\mathbb{E}\left[Wyy^\top W^\top\right] = (Cn\sigma^2 + \sigma^2\|\mathbb{E}\left[y\right]\|_2^2)I_m.$$

$\square$