1 Reviewer tESj (5;3):

Weaknesses

Limited Novelty - as far as I can tell Algorithm 1 is a direct combination of LAMB and Federated Averaging Weak Baslines - Since LAMB outperforms SGD and AMS, it seems expected that Fed-LAMB will outperform Fed-SGD and Fed-AMS. I feel like there should be comparisons with baselines like SCAFFOLD, especially for heterogeneous data where even if Fed-LAMB performs a bit worse than SCAFFOLD, the lower communcation cost as claimed in Section 3 (lines 234-243) might help make a case for it. Queries and Suggestions

Please clarify the key message in the discussion at the end of section 3. Specifically, in the first point on communication complexity it is claimed that Fed-LAMB has lower communication cost than SCAFFOLD. While this seems true, SCAFFOLD is designed to handle data heterogenity which Fed-LAMB doesn't seem explicitly designed to handle. Likewise the second point on homogeneous and heterogeneous data also seems to just describe how data heterogenity is handled in other works and not how the proposed approach can handle data heterogenity in theory. Moreover since there is no comparison with SCAFFOLD in the experiments, it is unclear if Fed-LAMB will perform comparable to SCAFFOLD in heterogeneous settings. Without this information or further explanation than what is currently given, the lower communication cost cannot be claimed as a benefit.

Is Fed-AMS same as local-AMS [2]? If not, then what is the difference between the two and why is there no comparison with local-AMS?

In lines 272 and 273 you say that :In each round, the training samples are allocated to the active devices..." Does this mean that training samples are moved across devices from round to round? Please clarify if that is the case and if so please provide a citation to justify this since the way I understand it, Federated Learning requires that the data remain on the device it belongs to.

Please provide an emprical comparison with SCAFFOLD if possible.

Our reply: – We thank the reviewer for the reference to SCAFFOLD. The difference is, that paper does SGD updates in each local worker with ADAM optimizer *applied to the global model*, while our method runs layerwise-adaptive AMSGrad at local workers with standard FedAvg for global aggregation. It would be interesting to compare these two different schemes. Thank you for the suggestion.

– Local-AMS is a distributed variant of AMSGrad. It does not tackle the privacy issue nor the communication bottleneck of federated learning. Besides, the update at each iteration is not scaled per layer of the network and hence its speed is supposedly worse. In this work, we focus on the settings where the data can not be transmitted to the central server (privacy) and particularly focus on multilayered neural network by adapting the update to the layers of the model.

– In the experiments, and to mimic the practical use of Federated Learning, we only activate a portion of the available workers. The data is allocated randomly to each device prior the training loop and the device selection and remains unchanged throughout the training. We will clarify that in the revision.

2 Reviewer vn4T (3;4):

Originality: This paper mainly builds off the work by Chen et al. [1] and slightly modifies local AMS by adding a layer-wise normalization in the local updates.

Quality: The empirical performance of the algorithm seems good. It is indeed a nice idea to set different hyper-parameters (e.g. learning rate) for different layers. However, I have several concerns on the motivation and the theoretical analysis:

The paper purports to accelerate training of deep neural networks by adding the layerwise normalization scheme. But this effect is not analyzed in the theory part.

Compared to Local AMS, the proposed algorithm saves no communication.

In assumption H4, needs to be upper bounded. But in line 139-140 the paper says can be identity. This is a contradiction.

In Assumption H4, how does influences the convergence?

Theorem 1 is not actually showing the convergence to stationary points. How large is the denominator ?

In Theorem 1, how large is the constant term?

In Corollary 1, there should be a constant term (constant sub-optimality).

Clarity: In Algo. 1, Line 8: what is the initialization of and ?

Theorem only established the convergence result for . Therefore, it would be more clear to remove the weight decay term in the algo box.

In section 2.2, there are some wrong cross-reference to the algorithm box. The authors may want to fix this in the future version.

Our reply:

As we state line 227, our method theoretically reduces the number of communication rounds needed to reach the same $\epsilon-$stationary point. We also recall that a $\epsilon-$stationary point is defined as the parameter θ where the suboptimality condition $\frac{1}{R}\sum_{r=1}^{R}\mathbb{E}\left[\left\|\frac{\nabla f(\overline{\theta_r})}{\sqrt{v_r^t}}\right\|^2\right]$ is lower than ϵ.

We will rewrite assumption H4 so that for any value $a \in \Theta$ the scale function is bounded from below and above. Here Θ is a closed subset of $\mathbb{R}_+^*$. Then, the identity function can easily be bounded on a closed set.

We initialize m as 0 and v as ϵ as stated in our algorithm 1.

While we simplify the notations in the Theorem by setting $\lambda = 0$, we include it in the algorithm for the purpose of generality, as practicioner tend to put a weight decay. Our rate can be obtained trivially with a non null decay and we will include it in the revision.

3 Reviewer ZxHK (4;4):

Originality: The paper takes an existing approach to federated learning, and replaces the optimizer in the inner loop with a different optimizer (using an adaptive optimizer in the inner loop was also done by Chen et al [2], this paper follows the same method). Thus while the new optimizer works better in experiments, it is not clear why this is a novel contribution. The given convergence result is similar to the one in Chen et al.

I would like to see a comparison of the accuracy vs the frequency of communication. Also, I would like to see experimental comparison against the server only implementation, to understand how much accuracy is being lost due to distributed computing.

Can you explain what you mean by a local epoch? In the algorithm, T is the number of steps, but in Theorem 1, T is referred to as a local epoch.

A few typos made the paper harder to understand: Line 238: "Yet, contrary to SCAFFOLD, our method only sends bits once per communication round": How many bits? Also the proposed method does send two vectors - the weights and the preconditioning vector. Figure 2: Caption refers to top line vs bottom, but there is only one row of figures. Section 2.2 needs careful proofreading. e.g. line 233 — "In particular, while Line 1 and Line 1 corresponds"

Our reply:

We thank the reviewer for the attention given to our work. Stressing on the novelty of our contribution, we want to say that in the context of Federated Learning, in particular in the *cross-device* settings with a large volume of devices, training deep neural networks is a burning challenge. Given the potentially cumbersome amount of data present in each device, being able to learn high dimensional and nonconvex models per device is of utmost importance. – The nature of our contribution is thus *to improve the local optimization method for each device* so that a better local model is learned in fewer iterations, in order to obtain *a natural improvement of the communication efficiency* of the overall federated method, hence requiring less rounds of communication to reach similar accuracy. Local AMS does not deal with such settings as it tackles the standard distributed optimization problem.

R denotes the number of communication round (outter loop) while T represents the number of iterations/gradient step (inner loop, hence the name local epoch) at each round R. This will be clarified.

Thank you for the typos, they will be fixed in the revision of our paper.

4 Reviewer hJz9 (4;4):

I have several concerns on the clarity and the theory of this paper, which are listed below.

In Algorithm 1, in each round, it seems that each device only computes the stochastic gradient once at step 6, and then reuses the same gradient for the local iterations? And the proof of Lemma 1 only considers 1 local iteration. In this case, there is no difference between performing local updates and mini-batch updates.

Corollary 1 only depicts the dependence on . However, it is clear that in assumption 2 is potentially much larger than (in fact, is always larger that).

I'm not sure how the convergence on would imply a convergence to the stationary points (convergence on). Could it be going to infinity that results in the convergence? Some discussion is needed.

I'm not sure what is the key benefits of Fed-LAMB. To my knowledge, LAMB is for large batch training for language models like bert. Does Fed-LAMB enjoy similar performance boost for these tasks? The experiments only consider small classification tasks. Actually, SGD with momentum is more popular for classification, as it always achieves higher test accuracies comparing to adam or adagrad.

The manuscript is not so clear and contains several typos: e.g., Line 133, 135, 173, 243.

For these reasons, I think this work is below the acceptance threshold.

Our reply:

We thank the reviewer for valuable comments. Below we address your concerns about our contribution.

We would like to clarify that all of our results are for multiple local iterations $T > 1$ (from Lemma 1 to Theorem 2). We are not sure where the reviewer has understood that it is not the case but we will make it even clearer in the revision. We will adapt the proof of Lemma 1 to reflect that).

In Algorithm 1, the stochastic gradient is computed after each local update of the local model, i.e. after each iteration local iteration t. This will be fixed in the revision, thank you for the pointer.

We would like to clarify on the last point addressed by the reviewer:

– **Purpose of FED-LAMB:** In the context of Federated Learning, in particular in the *cross-device* settings with a large volume of devices, training deep neural networks is a burning challenge. Given the potentially cumbersome amount of data present in each device, being able to learn high dimensional and nonconvex models per device is of utmost importance. – The nature of our contribution is thus *to improve the local optimization method for each device* so that a better local model is learned in fewer iterations, in order to obtain *a natural improvement of the communication efficiency* of the overall federated method, hence requiring less rounds of communication to reach similar accuracy.