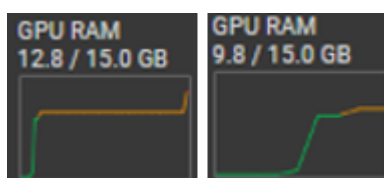


Hasil dari kedua model yang dilatih menggunakan dataset Liputan6 juga memiliki perkembangan *training loss* dan *validation loss* yang sama seperti kedua model yang dilatih dengan Indosum. Perbedaannya hanya ada di step ke 500 dimana terjadi *overfitting* untuk kedua model. Proses validasi kemudain mencegah terjadinya *overfitting* lagi di step setelah itu. Dari sisi sumber daya komputasi, hasilnya juga sama dengan kedua model yang dilatih oleh dataset Indosum, tetapi rasio berbanding pemakaian kedua model lebih besar di liputan6 dimana Bert memakai 12,8 GB dan DistilBert memakai 9,8 GB. Dapat disimpulkan bahwa efektifnya distilBert dalam mengurangi penggunaan sumberdaya system berkorelasi dengan besarnya dataset. Untuk pembelajaran menggunakan liputan6, durasi pembelajaran dari Bert adalah 3 jam 10 menit dan DistilBert memiliki durasi 1 jam 56 menit.



Gambar 8. Penggunaan RAM GPU pada saat proses training menggunakan dataset Liputan6 berjalan. Bert (a) dan DistilBert (b).

Dari hasil dari proses pembelajaran model Bert dan DistilBert, perkembangan *training loss*, baik yang dilatih dengan Indosum, maupun yang dilatih oleh Liputan6, mempunyai pola yang sama dengan Bert sedikit lebih cepat dalam penurunan nilai *training loss* untuk setiap 500 step. Walaupun ukuran model DistilBert adalah setengah dari besar model Bert, perkembangan DistilBert cukup dekat dengan Bert dengan menggunakan waktu dan sumber daya komputer lebih sedikit dibanding Bert.

## 5. Kesimpulan

Dalam evaluasi terhadap model DistilBert dan Bert, digunakan dua dataset berbeda yang memiliki ukuran dan kompleksitas yang berbeda. Hasilnya menunjukkan bahwa DistilBert menunjukkan tingkat kinerja yang mirip dengan model Bert, sementara menuntut sumber daya komputasi yang jauh lebih sedikit. Temuan ini menggarisbawahi efisiensi dan efektivitas DistilBert, menjadikannya pilihan untuk tugas-tugas pemrosesan bahasa alami dimana kendala komputasi dan waktu menjadi perhatian.

## Daftar Pustaka

- [1] Liu, Y., & Lapata, M. (2019). "Text summarization with pretrained encoders." arXiv preprint arXiv:1908.08345
- [2] F. Koto, A. Rahimi, J. H. Lau and T. Baldwin (2020). "Indolem and indobert: A benchmark dataset and pre-trained language model for indonesian nlp", *Proceedings of the 28th COLING*, 2020.
- [3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al. (2017). "Attention is all you need", *Proceedings of the 31st International Conference on Neural Information Processing Systems ser. NIPS'17*, pp. 6000-6010, 2017.
- [4] V. Sanh, L. Debut, J. Chaumond and T. Wolf, (2019). "Distilbert a distilled version of bert: smaller faster cheaper and lighter", *ArXiv*, vol. abs/1910.01108, 2019.
- [5] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova (2019). "BERT: Pretraining of deep bidirectional transformers for language understanding", *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies Volume 1 (Long and Short Papers)*, pp. 4171-4186, Jun. 2019.
- [6] K. Kurniawan and S. Louvan (2018). "Indosum: A new benchmark dataset for indonesian text summarization", *2018 International Conference on Asian Language Processing (IALP)*, pp. 215-220, 2018.
- [7] C. Wirawan (2020), *Indonesian bert2bert summarization model*, [online] Available: <https://huggingface.co/cahya/bert2bert-indonesian-summarization>.
- [8] C. Wirawan (2021), *Indonesian distilbert base model (uncased)*, [online] Available: <https://huggingface.co/cahya/distilbert-base-indonesian>.
- [9] Hang Li, Deep learning for natural language processing: advantages and challenges, *National Science Review*, Volume 5, Issue 1, January 2018, Pages 24–26
- [10] F. Koto, A. Rahimi, J. H. Lau and T. Baldwin (2020). "Indolem and indobert: A benchmark dataset and pre-trained language model for indonesian nlp", *Proceedings of the 28th COLING*, 2020.

[11] Fajri Koto, Jey Han Lau, and Timothy Baldwin. 2020. Liputan6: A Large-scale Indonesian Dataset for Text Summarization. In Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 598–608, Suzhou, China. Association for Computational Linguistics.

[12] Jackson, R. G., Patel, R., Jayatilake, N., Kolliakou, A., Ball, M., Gorrell, G., ... & Stewart, R. (2017). Natural language processing to extract symptoms of severe mental illness from clinical text: the Clinical Record Interactive Search Comprehensive Data Extraction (CRIS-CODE) project. *BMJ open*, 7(1), e012012.

[13] Gustavsson, A., Magnuson, A., Blomberg, B., Andersson, M., Halfvarson, J., & Tysk, C. (2013). On the difficulty of training recurrent neural networks. *Computer Science*, 52(3), 337-345.

[14] Sutskever, I. (2013). Training recurrent neural networks (pp. 1-101). Toronto, ON, Canada: University of Toronto.

### Lampiran

Kode yang dipakai untuk menjalankan percobaan dalam paper ini terdapat pada kedua link berikut:

Model memakai dataset Indosum:

<https://colab.research.google.com/drive/1mymEGqsAtLesfqiEcDktgoSNqqIOQz6k?usp=sharing>

Model memakai dataset Liputan6:

[https://colab.research.google.com/drive/1Pv2eAUOs\\_PFP\\_OqRjOe4mEzDjCNY6ZIk?usp=sharing](https://colab.research.google.com/drive/1Pv2eAUOs_PFP_OqRjOe4mEzDjCNY6ZIk?usp=sharing)

```

[ ] X1line
trainer.train()

[8919/8919 35:48, Epoch 1/1]

Step Training Loss Validation Loss
500 1.759800 1.593388
1000 1.698400 1.585483
1500 1.694100 1.563033
2000 1.671800 1.540833
2500 1.643500 1.551051
3000 1.644200 1.500393
3500 1.613200 1.501048
4000 1.611400 1.488188
4500 1.583300 1.468005
5000 1.574400 1.465582
5500 1.552300 1.451565
6000 1.542400 1.438181
6500 1.525900 1.414547
7000 1.526900 1.411152
7500 1.506300 1.401281
8000 1.488800 1.393019
8500 1.479500 1.380055

CPU times: user 35min 55s, sys: 4.71 s, total: 36min
Wall time: 36min 49s
TrainOutput(global_step=8919, training_loss=1.589424417384835, metrics={'train_runtime': 2148.3856, 'train_samples_per_second': 33.212, 'train_steps_per_second': 4.152, 'total_flos': 943316988998224.8, 'train_loss': 1.589424417384835, 'epoch': 1.0})
    
```

```

[ ] X1line
trainer.train()

[8919/8919 20:47, Epoch 1/1]

Step Training Loss Validation Loss
500 2.054800 1.835738
1000 2.008500 1.910206
1500 1.998100 1.883382
2000 1.978900 1.868754
2500 1.978300 1.858177
3000 1.957000 1.818384
3500 1.932400 1.799248
4000 1.908900 1.789675
4500 1.889600 1.771027
5000 1.875000 1.779007
5500 1.862000 1.763844
6000 1.844100 1.749781
6500 1.828200 1.724519
7000 1.832100 1.738001
7500 1.821000 1.707989
8000 1.802200 1.695944
8500 1.798200 1.692588

CPU times: user 28min 52s, sys: 3.95 s, total: 29min 56s
Wall time: 28min 49s
TrainOutput(global_step=8919, training_loss=1.898928792288353, metrics={'train_runtime': 1247.9129, 'train_samples_per_second': 57.175, 'train_steps_per_second': 7.147, 'total_flos': 4729281223865688.0, 'train_loss': 1.898928792288353, 'epoch': 1.0})
    
```