Nama : Andreas Hasiholan Sinaga

NIM : 1103213033

RNN dan Deep RNN model

Recurrent Neural Networks (RNN) adalah jenis jaringan neural yang dirancang untuk memproses data berurutan, seperti teks, sinyal waktu, atau data serial lainnya. Berbeda dengan jaringan neural biasa, RNN memiliki *feedback loop* yang memungkinkan informasi dari langkah sebelumnya digunakan untuk memengaruhi langkah berikutnya, sehingga jaringan ini memiliki kemampuan untuk "mengingat" informasi sebelumnya dalam urutan data. Namun, RNN standar sering menghadapi kendala, seperti *vanishing gradient problem*, yang membuatnya kurang efektif dalam menangkap hubungan jangka panjang pada data yang memiliki urutan panjang.

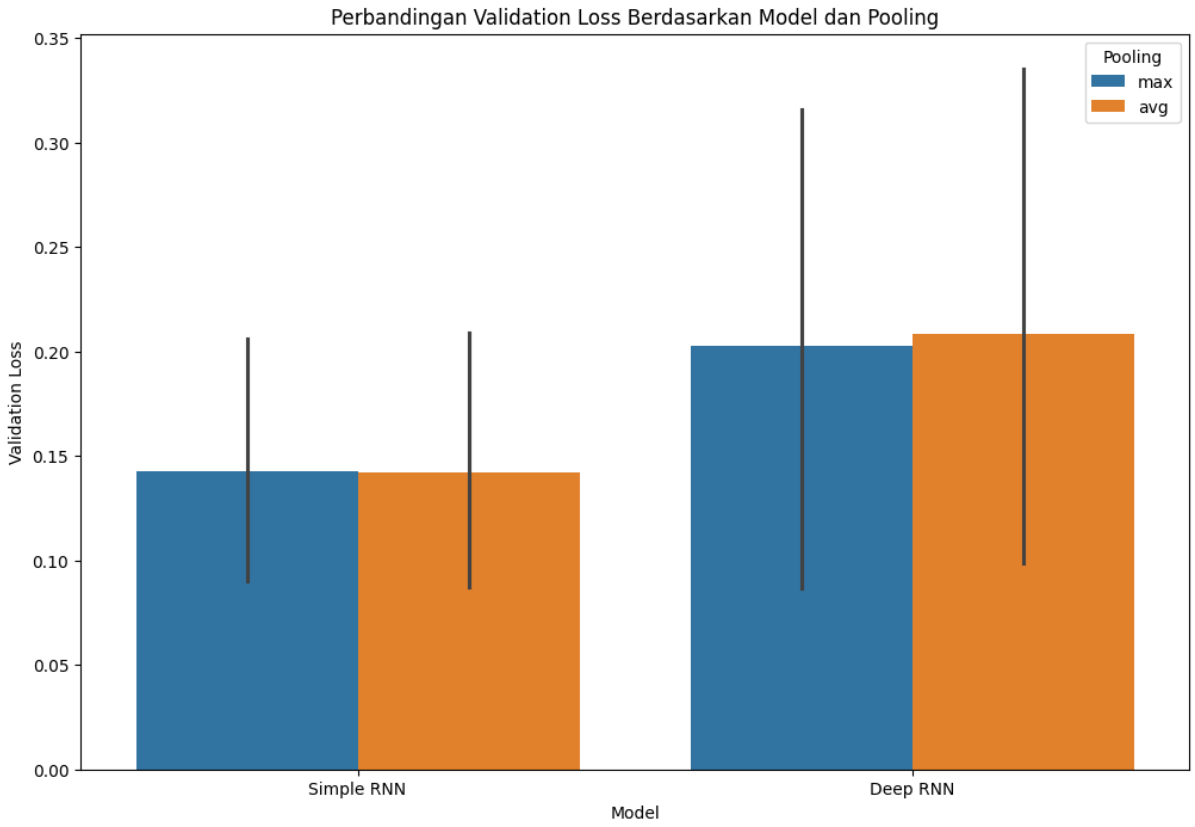
Untuk mengatasi keterbatasan ini, Deep Recurrent Neural Networks (Deep RNN) diperkenalkan sebagai pengembangan dari RNN standar dengan menambahkan beberapa lapisan tersembunyi (*hidden layers*) pada arsitekturnya. Struktur bertumpuk ini memungkinkan Deep RNN untuk menangkap pola yang lebih kompleks dan hubungan hierarkis dalam data. Dengan demikian, Deep RNN lebih efektif untuk menangani tugas-tugas yang melibatkan data dengan kompleksitas tinggi atau urutan panjang, seperti *language modeling* tingkat lanjut, analisis video, dan pemrosesan suara. Meskipun memiliki keunggulan ini, Deep RNN juga membutuhkan lebih banyak sumber daya komputasi dibandingkan RNN standar. Untuk mengatasi masalah *vanishing gradient* pada kedua jenis jaringan ini, varian seperti Long Short-Term Memory (LSTM) dan Gated Recurrent Unit (GRU) sering digunakan, karena mampu mempertahankan informasi lebih baik dalam urutan data yang panjang.

```
Hasil Terbaik:
Model: Deep RNN, Pooling: max, Optimizer: RMSProp, Epochs: 350, Validation Loss: 0.0351
Model: Deep RNN, Pooling: max, Optimizer: RMSProp, Epochs: 50, Validation Loss: 0.0379
Model: Deep RNN, Pooling: max, Optimizer: Adam, Epochs: 50, Validation Loss: 0.0379
Model: Deep RNN, Pooling: max, Optimizer: RMSProp, Epochs: 250, Validation Loss: 0.0380
Model: Deep RNN, Pooling: max, Optimizer: RMSProp, Epochs: 100, Validation Loss: 0.0381
```

Hasil pemodelan menggunakan Deep Recurrent Neural Network (Deep RNN) menunjukkan bahwa kombinasi parameter tertentu menghasilkan performa yang berbeda, diukur berdasarkan *validation loss*. Lima konfigurasi terbaik yang diperoleh semuanya menggunakan *max pooling*, yang menunjukkan efektivitas metode ini dalam menangkap fitur penting dari data. Kombinasi terbaik dicapai dengan *optimizer* RMSProp dan jumlah *epochs* sebanyak 350, menghasilkan *validation loss* terendah, yaitu 0.0351. Hal ini menunjukkan bahwa pelatihan dengan jumlah *epochs* yang lebih panjang, didukung oleh stabilitas RMSProp, memungkinkan model mempelajari pola dalam data secara lebih baik. Kombinasi RMSProp dengan jumlah *epochs* lebih sedikit, seperti 50 dan 250, menghasilkan *validation loss* masing-masing sebesar 0.0379 dan 0.0380.

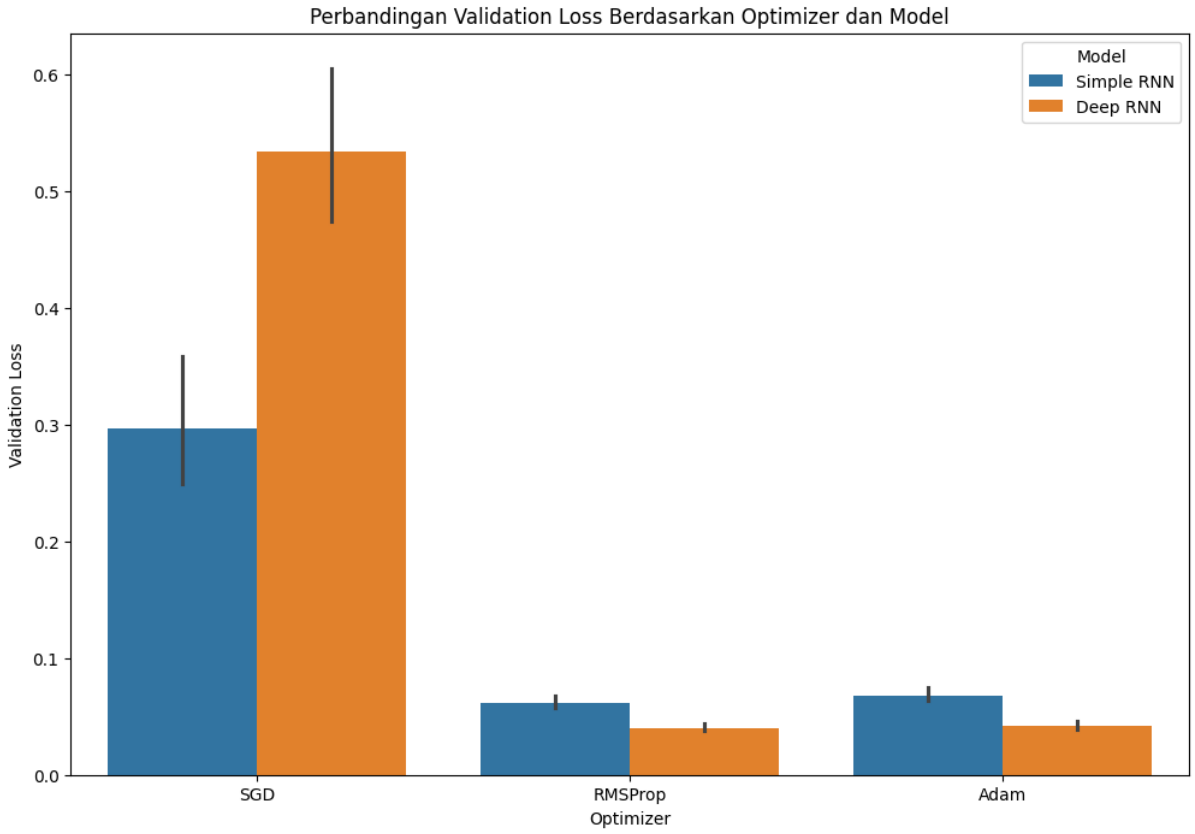
Selain itu, penggunaan *optimizer* Adam dengan jumlah *epochs* sebanyak 50 juga menghasilkan *validation loss* sebesar 0.0379, yang setara dengan hasil menggunakan RMSProp pada jumlah *epochs* yang sama. Meskipun demikian, RMSProp cenderung lebih unggul dalam pelatihan

dengan jumlah *epochs* yang lebih panjang. Kombinasi dengan jumlah *epochs* yang lebih sedikit, seperti 100, menghasilkan *validation loss* sedikit lebih tinggi, yaitu 0.0381. Secara keseluruhan, kombinasi *optimizer* RMSProp dengan *max pooling* dan jumlah *epochs* yang cukup panjang memberikan hasil terbaik, tetapi kombinasi dengan jumlah *epochs* lebih sedikit atau menggunakan *Adam optimizer* dapat menjadi alternatif yang efisien dalam kondisi tertentu.



Grafik di atas menunjukkan perbandingan *validation loss* berdasarkan jenis model (Simple RNN dan Deep RNN) serta metode *pooling* yang digunakan (*max pooling* dan *average pooling*). Pada Simple RNN, metode *max pooling* dan *average pooling* menghasilkan *validation loss* yang relatif rendah, dengan perbedaan yang tidak terlalu signifikan. Namun, pada Deep RNN, terdapat pola serupa di mana *max pooling* menghasilkan performa yang lebih baik dibandingkan *average pooling*, meskipun perbedaannya masih dalam rentang yang wajar.

Dari grafik tersebut, terlihat bahwa Deep RNN secara umum memiliki rentang *validation loss* yang lebih besar dibandingkan Simple RNN, mengindikasikan bahwa model ini lebih kompleks dan sensitif terhadap variasi dalam data. Namun, performa terbaik diperoleh pada Deep RNN dengan *max pooling*, yang menunjukkan bahwa kombinasi ini lebih efektif dalam mengekstraksi fitur dari data. Hasil ini mendukung penggunaan Deep RNN dengan *max pooling* sebagai konfigurasi yang optimal untuk tugas-tugas dengan pola data yang lebih kompleks.



Grafik di atas menunjukkan perbandingan *validation loss* berdasarkan jenis *optimizer* (SGD, RMSProp, dan Adam) dan model yang digunakan (Simple RNN dan Deep RNN). Dari hasil ini, terlihat bahwa penggunaan *optimizer* RMSProp dan Adam menghasilkan *validation loss* yang lebih rendah dibandingkan SGD pada kedua jenis model. Hal ini menunjukkan bahwa RMSProp dan Adam lebih efektif untuk pelatihan model dalam konteks data yang digunakan, karena mampu menangani masalah optimasi dengan lebih baik, seperti *vanishing gradient*.

Pada Simple RNN, RMSProp memberikan *validation loss* terendah, diikuti oleh Adam, sedangkan SGD menunjukkan performa yang lebih buruk dengan *validation loss* yang lebih tinggi. Pola serupa juga terlihat pada Deep RNN, di mana RMSProp tetap menjadi *optimizer* yang paling unggul, diikuti oleh Adam. Namun, Deep RNN dengan SGD memiliki *validation loss* yang jauh lebih tinggi dibandingkan Simple RNN dengan SGD, menunjukkan bahwa Deep RNN lebih sensitif terhadap pemilihan *optimizer*.

Secara keseluruhan, hasil ini menegaskan pentingnya pemilihan *optimizer* yang tepat untuk mencapai performa terbaik pada model. RMSProp dan Adam adalah pilihan yang sangat baik untuk mengoptimalkan performa, terutama pada model yang lebih kompleks seperti Deep RNN. Kombinasi Deep RNN dengan RMSProp memberikan hasil yang paling menjanjikan dalam konteks ini