STRING (Satuan Tulisan Riset dan Inovasi Teknologi) p-ISSN: 2527 - 9661
Vol. 8 No. 1 Agustus 2023 e-ISSN: 2549 - 2837

PERBANDINGAN OPTIMIZER ADAGRAD, ADADELTA DAN ADAM DALAM KLASIFIKASI GAMBAR MENGGUNAKAN DEEP LEARNING

Shedriko[1], Muhammad Firdaus[2]
Program Studi Teknik Informatika, Universitas Indraprasta PGRI[1,2]
shedriko@gmail.com[1], dasurichi@gmail.com[2]

Submitted February 19, 2023; Revised July 16, 2023; Accepted July 30, 2023

Abstrak

Teknologi pengenalan gambar berkembang sangat cepat beberapa waktu terakhir. Banyak metodologi bermunculan dalam penggunaannya. Salah satunya adalah *Convolutional Neural Network* (CNN) yang digunakan pada penelitian ini. Metodologi tersebut digunakan untuk mendeteksi gambar pola dari bentuk susunan jari satu tangan sebagai isyarat dari identifikasi angka 0 sampai 9 dalam SIBI (Sistem Bahasa Isyarat Indonesia). Permasalahan dari penelitian ini adalah banyak optimizer bermunculan dalam metodologi *deep learning*, sehingga pemilihan optimizer yang tepat menjadi tantangan tersendiri untuk dijadikan acuan selanjutnya dari gambar *input* yang tidak melalui tahap *pre-processing* sebelumnya. Tujuan penelitian adalah mendapatkan nilai akurasi yang terbaik dari perbandingan 3 *optimizer* serta keterkaitannya dengan waktu proses. Kesimpulan yang diperoleh menunjukkan bahwa optimizer AdaDelta yang meskipun telah lama muncul dapat memberikan hasil yang lebih baik dari Adam yang merupakan perkembangan dari *optimizer* terakhir.
Kata Kunci : CNN, *deep-learning*, klasifikasi, *optimizer*.

Abstract

Image recognition technology has developed rapidly in recent times. There are many methods springing up in its use. One of them is the Convolutional Neural Network (CNN) as used in this research. The method is used to detect image patterns from the shape of the arrangement of the fingers of one hand as a signal from the identification of the numbers 0 to 9 in SIBI (Indonesian Sign Language System). The problem of the research is that many optimizers emerge in a deep learning method. Therefore, selecting the right optimizer itself is a challenge that can be used as the next reference for input images that do not go through the previous pre-processing stage. The aim of the research is to get the best accuracy score from the comparison of 3 optimizers and their relations to processing time. The conclusion obtained shows that AdaDelta optimizer that has existed for a long time can provide better results than Adam which is the development of the last optimizer.
Keywords : *CNN, deep-learning, classification, optimizer*.

1. PENDAHULUAN

Beberapa tahun terakhir, metodologi *deep learning* telah menjadi primadona dan dijadikan standard emas dalam komunitas *machine learning* [1]. Metodologi tersebut diimplemetasikan dalam *Convolutional Neural Network* (CNN) yang memberikan gambaran sama dengan cara kerja *core sensorial region* pada otak manusia. CNN menggunakan pola berlapis dalam mengklasifikasi suatu objek gambar dari *input* dengan data gambar yang telah di*train* sehingga memberikan kemiripan *output* dengan gambar *input*. CNN menggunakan faktor *optimizer* untuk memaksimalkan proses klasifikasi, yang dalam perkembangannya beberapa jenis *optimizer* telah dirumuskan oleh para peneliti terdahulu. Permasalahan dari penelitian ini adalah membandingkan beberapa optimizer, terutama optimizer yang dalam kurun waktu belakangan ini cukup populer, seperti AdaGrad

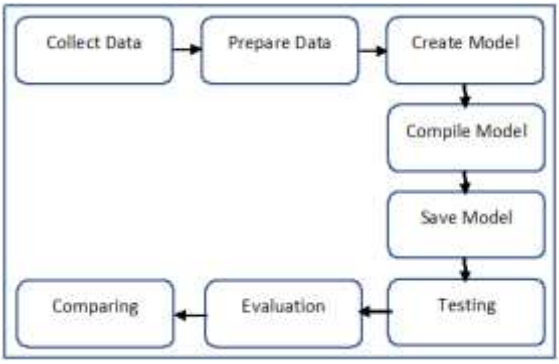
(*Adaptive Gradient*), AdaDelta (*Adaptive Delta*) dan Adam (*Adaptive Moment Estimation*). Dalam klasifikasi gambar, *issue* umum yang ada adalah keberadaan tahap *pre-processing* dari gambar *input*. Pada penelitian ini proses klasifikasi dilakukan tanpa merubah kondisi apapun dari gambar *input* atau dilakukan tanpa *pre-processing* [2]. Tahap tanpa *pre-processing* ini menjadi kondisi penyerta dari permasalahan dalam penelitian ini. Sedangkan tujuan dari penelitian ini adalah mendapatkan nilai akurasi terbaik dari perbandingan optimizer AdaGrad [3], AdaDelta [4] dan Adam [5] dalam kurun waktu proses tertentu.

Alasan diadakan penelitian ini adalah mendapatkan informasi yang lebih baik mengenai identifikasi gambar dengan menggunakan metodologi pada penelitian ini. Tujuan dilakukan penelitian adalah untuk mendapatkan perbandingan *optimizer* terbaik dalam mengindentifikasi gambar dengan metodologi tersebut.

2. METODE PENELITIAN

Penelitian ini menggunakan metode A*pplied* atau Aplikatif untuk penyelesaian tulisan secara keseluruhan [6]. Metode ini bertujuan untuk mencari solusi langsung dalam menyelesaikan masalah industri, dalam hal ini berupa melakukan perbandingan menggunakan Bahasa Pemrograman Python dengan *tools* Jupyter Notebook. Bahasa pemograman dengan *code* programnya yang simple dan mudah dimengerti serta *powerfull* ini [7] menggunakan library Keras sebagai API (*Application Programming Interface*) *deep learning* yang berjalan di atas platform *machine learning* yaitu Tensorflow [8]. Metodologi yang digunakan adalah Convolutional Neural Network (CNN). Untuk klasifikasi menggunakan library numpy dan cv2, serta sklearn yang digunakan untuk mengumpulkan *classification report*.

Alur dalam mengerjakan penelitian adalah sebagai berikut:



Gambar 1. Alur Metodologi Penelitian

Collect Data. Pada tahapan ini penulis mengumpulkan berbagai data yang terkait dengan gambar posisi jari pada satu tangan yang memberikan arti isyarat dari 0 sampai 9 dalam SIBI [9]. Data yang dipilih merupakan data yang jelas dan dapat dipahami oleh tatapan mata normal pada umumnya. Hal tersebut untuk mengurangi *error* atau *miss-interpretation* yang mungkin terjadi, baik pada saat *training* maupun *testing* data.

Prepare Data. Sebanyak 2.062 gambar berhasil dikumpulkan yang kemudian dikelompokkan ke dalam 10 kelas yang terdiri dari 0 sampai 9. Dalam penyiapan data, dibuat data *augmentation* dari jumlah gambar tersebut dengan sintaks sebagai berikut:

```
train_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.1,horizontal_flip=False,validation_split= 0.2)
```

Penskalaan dibuat hingga 255 sebagai acuan dari *range* tingkat keterangan gambar dari 0 hingga 255. Jumlah gambar dibagi 80% untuk di*train* dan 20% untuk dievaluasi seperti yang terlihat pada *validation_split* sebesar 0,2. Kemudian *train_datagen* dikelompokkan ke dalam *subset training* menjadi *train_gen* dan *subset validation* menjadi *val_gen* yang

keduanya menjadi parameter input untuk fase *compile* model.

Create Model. Model dibuat secara *sequential* dengan sintaks konvolusi sebagai berikut:

```
model.add(Conv2D(32,kernel_size=3,strides=1,activation='relu',padding='same',input_shape=(100,100,3)))
```

Konvolusi dibuat dengan jumlah filter 32 dengan ukuran kernel 3x3, jarak pergeseran sebesar satu, aktivasi menggunakan RELU (*Rectified Linear Unit*) dengan output yang sama, lalu dimensi gambar input ditentukan 100x100 menggunakan 3 *channel*. Diikuti dengan sintaks sebagai berikut:

```
model.add(MaxPooling2D(pool_size=(3,3),strides=2))
```

Menggunakan *pooling* nilai maximum dengan ukuran 3x3 dengan pergeseran sebesar dua. Kedua sintaks tersebut dibuat berulang sebanyak 3x dengan jumlah filter secara berurutan adalah 64, 64 dan 128. Pergeseran *pooling* sama kecuali yang terakhir dibuat sebesar satu. Kemudian diikuti dengan tahap *flatten*, *drop out* serta *dense*. Pada *model.compile* parameter pertama *optimizer* diisi dengan "AdaGrad".

Compile Model. Fase ini merupakan penentuan dari jumlah *epoch* atau iterasi yang mewakili. Dengan melihat nilai *accuracy* pada *epoch* yang bernilai lebih dari 90% maka jumlah *epoch* ditentukan sebanyak 10x perulangan. Sintaks yang digunakan adalah:

```
model.fit_generator(train_gen,epochs=10,validation_data=val_gen)
```

dimaana penggunaan parameter standar dari python dengan kolom isian dinamis hanya pada parameter jumlah *epoch* yang diisikan.

Save Model. Dilakukan penyimpanan model dalam bentuk file json dengan fungsi *write* sehingga dapat digunakan berulang-ulang.

Testing. Proses *load* dari model yang telah disimpan dilakukan terlebih dulu. Kemudian mengambil salah satu gambar yang akan di*test* dengan sintaks sebagai berikut:

```
image = cv2.imread()
```

dimana tanda kurung diisi dengan lokasi dari gambar input yang akan di*test*. Lalu dilakukan predikasi dengan sintaks:

```
prediksi = load_model.predict_classes(image)
```

Sebelum dan sesudah sintaks prediksi diselipkan sintaks penghitungan waktu proses, dimana sebelum eksekusi sintaks prediksi digunakan parameter "tic" sedangkan setelahnya menggunakan parameter "toc" dengan sintaks yang sama sebagai berikut:

```
tic = time.process_time()
```

Evaluation. Penilaian dari kinerja model dilakukan dengan men*generate classification report*. Nilai *precision* dari tiap gambar isyarat serta nilai akurasi dari pengetesan gambar *input* ditampilkan pada fase ini. Nilai-nilai tersebut diperolah dari rumusan faktor *true* dan *false* dari nilai positif dan negatifnya. Demikian pula untuk nilai *recall* yang ditampilkan. Sedangkan faktor *f1-score* merupakan nilai yang menunjukkan baik atau buruknya suatu data terhadap nilai *precision* dan *recall*nya pada pengetesan dari data yang dipilih [10].

Comparing. Fase ini membandingkan kondisi dari parameter *optimizer* pertama AdaGrad dengan AdaDelta dan Adam pada *optimizer* lainnya. Pengerjaan dilakukan dengan mengulang proses pada tahap *create model* sampai *evaluation* dengan masing-masing *optimizer* lainnya tersebut. Setiap selesai menjalankan satu *optimizer*, maka jupyter notebook akan dikeluarkan dari sistem untuk kemudian dijalankan ulang dari permulaan. Hal tersebut untuk menghindari penambahan iterasi sebelumnya yang membutuhkan memory cukup banyak pada sistem komputer, sehingga dengan me*restart tools* tersebut akan me*refresh* sistem pada langkah

selanjutnya tanpa adanya pengaruh dari proses *epoch* sebelumnya.

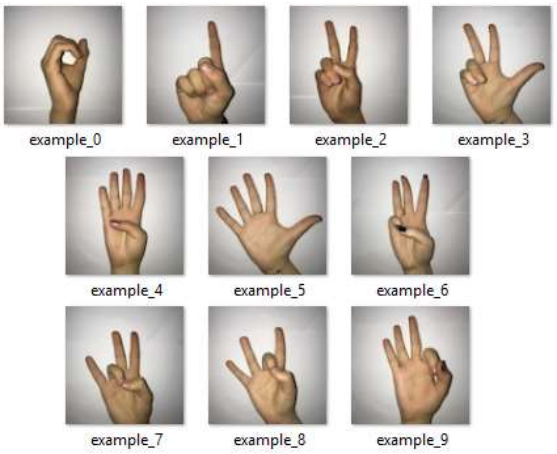
3. HASIL DAN PEMBAHASAN

Sejumlah 2.062 gambar yang mewakili lambang isyarat jari pada satu tangan dari 0 sampai 9 diambil sebagai master data. 80% dari jumlah tersebut dit*rain* sebagai data acuan untuk klasifikasi. Berikut beberapa contoh gambar dari isyarat angka tersebut :

Gambar 2. Beberapa contoh isyarat dari angka 3 (tiga)

Gambar 3. Beberapa contoh isyarat dari angka 6 (enam)

Dari keseluruhan dataset, sebanyak 1.653 gambar digunakan untuk data *train* sementara 409 gambar lainnya digunakan untuk evaluasi. Gambar yang diambil terdiri dari berbagai sumber dengan objek yang berbeda namun dengan posisi isyarat yang sama. Dataset dikumpulkan dalam folder yang sama dengan program berbahasa Python menggunakan *tools* Jupyter Notebook sehingga menghasilkan file berekstensi ipynb. *Epoch* ditentukan sebanyak 10x putaran dengan pertimbangan dari beberapa pengetesan yang menunjukkan bahwa tidak semua *optimizer* mencapai akurasi iterasi hingga lebih dari 90% dalam epoch yang kurang dari 5x putaran. Berikut adalah gambar test yang digunakan:



Gambar 4. Gambar *Test*

Model pertama dibuat dengan *optimizer* AdaGrad. *Optimizer* ini mencapai nilai akurasi ierasi yang lebih dari 90% pada *epoch* ke 7 sebesar 0,9298. Selanjutnya angka tersebut stabil hingga akhir *epoch*. Untuk angka *precision*, *recall* dan *f1-score* dari AdaGrad dapat dilihat pada tabel berikut:

Tabel 1. *Precision, Recall* dan *F1-Score* dari AdaGrad

Number	Precision	Recall	F1-Score
0	0,14	0,25	0,18
1	0,69	0,45	0,55
2	0,14	0,10	0,12
3	0,07	0,05	0,06
4	0,03	0,05	0,03
5	1,00	0,20	0,33
6	0,25	0,75	0,38
7	0,07	0,05	0,06
8	0,00	0,00	0,00
9	1,00	0,05	0,10

Pada *optimizer* AdaGrad, nampak bahwa *precision* tertinggi diperlihatkan pada klasifikasi angka 5 dan 9 dengan nilai 1, sementara yang terendah ada pada angka 8. Untuk *recall*, nilai tertinggi terlihat pada angka 6 dan nilai terendahnya juga pada angka 8 dengan nilai 0,00. Nilai *f1-score* 0,00 pada angka 8 menunjukkan nilai *precision* dan *recall* yang terburuk.

Kemudian pada optimizer AdaDelta, nilai akurasi iterasi 90% diperoleh pada iterasi ke 4 dengan nilai 0,9141. Setelah itu nilai akurasi berjalan stabil di atas nilai tersebut hingga *epoch* ke 10. Untuk nilai *precision*, *recall* dan *f1-score* dari *optimizer* AdaDelta dapat dilihat pada tabel berikut.

Tabel 2. *Precision, Recall* dan *F1-Score* dari AdaDelta

Number	Precision	Recall	F1-Score
0	1,00	0,75	0,86
1	0,17	0,05	0,08
2	0,11	0,05	0,07
3	0,16	0,25	0,20
4	0,19	0,25	0,22
5	0,44	0,55	0,49
6	0,32	0,55	0,41
7	0,07	0,05	0,06
8	0,00	0,00	0,00
9	0,04	0,05	0,04

Nilai *precision* tertinggi dari optimizer AdaDelta terlihat pada pengetesan dari angka 0, dengan nilai 1. Sedangkan nilai terendah ada pada angka 8 dengan nilai 0,00. Nilai *recall* tertinggi berada pada angka 0 sedangkan yang terburuk juga berada pada angka 8. Dari nilai *precision* dan *recall* tersebut dapat disimpulkan pada nilai *f1-score*nya bahwa nilai terburuk keduanya ada pada angka 8.

Selanjutnya pada optimizer Adam, nilai akurasi iterasi di atas 90% berada pada epoch ke 4 dengan nilai 0,9250. Sampai epoch ke 10 nilai akurasi iterasi stabil di atas 90%. Nilai *precision*, *recall* dan *f1-score* nya dapat dilihat pada tabel berikut.

Tabel 3. *Precision, Recall* dan *F1-Score* dari Adam

Number	Precision	Recall	F1-Score
0	0,26	0,80	0,40
1	0,50	0,25	0,33
2	0,17	0,05	0,08
3	0,05	0,05	0,05
4	0,00	0,00	0,00
5	0,22	0,20	0,21
6	0,38	0,75	0,51
7	0,09	0,05	0,06
8	0,00	0,00	0,00
9	0,50	0,05	0,09

Angka 1 dan 9 memberikan nilai *precision* tertinggi, yaitu masing-masing bernilai 0,5. Nilai *recall* tertinggi terlihat pada angka 0 dengan nilai 0,8. Nilai terburuk keduanya nampak pada angka 4 dan 8 dengan nilai 0,00. Nilai tersebut terlihat pada nilai *f1-score*nya yang juga bernilai 0,00.

Nilai 0,00 pada *f1-score*, yang berarti juga terkait dengan nilai *precision* dan *recall*nya, menunjukkan adanya nilai 0, baik terhadap *true positive* (TP) dan *false positive*nya (FP), maupun terhadap *true negative* (TN) dan *false negative*nya (FN). Karena dari sintaks program yang digunakan tidak menampilkan nilai-nilai tersebut, maka tidak dapat diketahui apakah nilai 0,00 terkait dengan prediksi keseluruhan negatif atau keseluruhan positif. Hal ini menjadi kelemahan dari sintaks yang digunakan pada penelitian ini. Namun di luar kondisi tersebut, dari perbandingan ketiga tabel di atas menunjukkan bahwa optimizer Adam memiliki 2 data dengan *f1-score* yang bernilai 0,00 , sedangkan AdaGrad dan AdaDelta masing-masing hanya memiliki 1 datum dengan nilai terburuk tersebut. Hal ini menunjukkan bahwa *optimizer* AdaGrad dan AdaDelta memberikan

prediksi nilai *precision* dan *recall* yang lebih baik dibanding Adam.

Fitur *classication report* juga memberikan nilai *accuracy* dari *f1-score*. Nilai akurasi yang diberikan merupakan nilai akurasi rata-rata dari keseluruhan data yang diprediksi. Penghitungan terhadap *processing time* juga dilakukan untuk melihat kaitannya dengan nilai akurasi. Nilai accuracy dan processing time dari masing-masing optimizer dapat dilihat pada tabel berikut.

Tabel 4. Nilai *Accuracy* rata-rata dan *Processing Time* dari masing-masing *optimizer*

	Proc-Time (s)	Accuracy
AdaGrad	0,109375	0,2
AdaDelta	0,078125	0,26
Adam	0,0625	0,22

Dari tabel 4 terlihat bahwa AdaGrad memiliki nilai *accuracy* rata-rata sebesar 0,2 dengan waktu proses yang dibutuhkan adalah 0,10935 detik. Nilai 20% merupakan tingkat nilai yang cukup rendah untuk suatu akurasi. Namun bila disandingkan dengan waktu prosesnya, terbilang sangat cepat karena berada pada kisaran di bawah 1 detik.

Optimizer AdaDelta memiliki nilai akurasi rata-rata 0,26 dengan waktu proses 0,078125 detik. Dengan nilai waktu yang berada di bawah 0,01, processing time dari optimizer ini tergolong luar biasa cepat. Sementara nilai akurasinya masih berada pada kisaran rendah karena berada di bawah 30%.

Sedangkan *optimizer* Adam memiliki nilai akurasi rata-rata sebesar 0,22 dengan waktu proses adalah 0,0625 detik. Waktu proses yang dihasilkan juga terbilang luar biasa cepat dengan kisaran yang berada di bawah 0,01 detik. Adapun nilai akurasi masih berada pada kisaran rendah karena juga masih berada pada nilai kurang dari 30%.

Dari hasil perbandingan ketiga *optimizer* terlihat bahwa AdaDelta memiliki nilai akurasi yang paling baik dibanding dua *optimizer* lainnya. AdaDelta unggul dengan selisih nilai akurasi rata-rata 0,04 atau 4%, namun memiliki waktu proses yang lebih panjang sekitar 0,015625 detik dibandingkan dengan Adam. Nilai akurasi kerap berbanding terbalik dengan waktu proses. Semakin baik nilai akurasinya maka akan semakin lambat waktu prosesnya. Sedangkan AdaGrad berada pada urutan paling bawah dengan nilai akurasi rata-ratanya adalah 0,2 atau 20% dan waktu proses paling lama di kisaran 0,109375 detik.

Nilai akurasi rendah yang ditunjukkan pada penelitian ini menguatkan para peneliti sebelumnya tentang issue dari pengenalan atau klasifikasi gambar bahwa gambar input yang tidak dilakukan *pre-processing* akan memberikan nilai akurasi yang rendah dengan *processing time* yang relatif sangat cepat. Hal tersebut berlaku sebaliknya terhadap gambar input yang dilakukan *pre-processing* terlebih dulu.

Penilaian yang mencakup akurasi rata-rata dan waktu proses, menunjukkan bahwa *optimizer* AdaDelta memberikan hasil yang lebih baik dibandingkan dengan Adam. Nilai akurasi yang lebih baik dengan sedikit lebih lambat dalam waktu proses menunjukkan bahwa AdaDelta masih sangat dapat diandalkan meskipun harus dibandingkan dengan Adam yang merupakan *optimizer trend* terkini dalam *deep learning*.

4. SIMPULAN

Beberapa hal yang dapat disimpulkan dari penelitan ini adalah bahwa AdaGrad memerlukan iterasi ke 7 untuk mendapat nilai akurasi itersi di atas 90%, sedangkan AdaDelta dan Adam pada iterasi ke 4.

Adam memiliki 2 nilai dengan *f1-score* terendah, sedangkan AdaGrad dan AdaDelta masing-masing hanya 1 nilai. AdaDelta memberikan nilai akurasi terbaik, sementara Adam memberikan nilai waktu proses tercepat.

Sintax yang digunakan tidak dapat menampilkan nilai TP, TN, FP dan FN sehingga tidak dapat memprediksi keseluruhan positif atau negatif dari nilai *f1-score* sama dengan 0 (nol). Nilai akurasi prediksi rata-rata yang kecil pada ketiga *optimizer* tersebut menunjukkan tidak adanya *pre-processing* pada gambar input.

Ketiadaan *pre-processing* pada gambar input menyebabkan waktu proses yang sangat luar biasa cepat pada ketiga *optimizer* tersebut. Metode atau *optimizer* terbaru atau yang sedang *trend* tidak menjamin memberikan hasil yang terbaik untuk setiap kasus pada *deep learning*

DAFTAR PUSTAKA

[1] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J. Big Data, Springer Open*, vol. 8, no. 53, 2021.

[2] G. Ranganathan, "A Study to Find Facts Behind Preprocessing On Deep Learning Algorithms," *J. Innov. Image Process.*, vol. 03, no. 01, pp. 66–74, 2021.

[3] A. A. Lydia et al, "Adagrad - An Optimizer for Stochastic Gradient Descent," *Int. J. Inf. Comput. Sci.*, vol. 6, no. 5, 2019.

[4] Z. Qu et al, "Genetic Optimization Method of Pantograph and Catenary Comprehensive Monitor Status Prediction Model Based on Adadelta Deep Neural Network," *IEEE Access*, vol. 7, 2019.

[5] S. Bock, "An improvement of the convergence proof of the ADAM-optimizer," *Conf. Pap. Oth Clust.*, 2018.

[6] C. R. Kothari, *Research Methodology, Methods & Techniques*, Second Rev. New Age International (P) Ltd., 2004.

[7] E. Willow, *OpenCV-Python Tutorials Documentation, Release Beta*. OpenCV, 2017.

[8] M. Abadi, "A Computational Model for TensorFlow," *Proceeding 1st ACM SIGPLAN Int. Work. Mach. Learn. Program. Lang.*, 2017.

[9] M. B. S. Bakti and Y. M. Pranoto, "Pengenalan Angka Sistem Isyarat Bahasa Indonesia Dengan Menggunakan Metode Convolutional Neural Network," *Semin. Nas. Inov. Teknol.*, 2019.

[10] R. Yacouby and D. Axman, *Probabilistic Extension of Precision, Recall, and F1 Score for More Thorough Evaluation of Classification Models*. Association for Computational Linguistics, 2020.