DAFTAR PUSTAKA

Ahuja, M.K., Singh, A., 2015. Static vision based Hand Gesture recognition using principal component analysis, in: 2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE). Presented at the 2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE), pp. 402–406. https://doi.org/10.1109/MITE.2015.7375353

Chuan, C., Regina, E., Guardino, C., 2014. American Sign Language Recognition Using Leap Motion Sensor, in: 2014 13th International Conference on Machine Learning and Applications. Presented at the 2014 13th International Conference on Machine Learning and Applications, pp. 541–544. https://doi.org/10.1109/ICMLA.2014.110

Dzulkarnain, I., Sumpeno, S., Christyowidiasmoro, C., 2016. Pengenalan Isyarat Tangan Menggunakan Leap Motion Controller untuk Pertunjukan Boneka Tangan Virtual. Jurnal Teknik ITS 5. https://doi.org/10.12962/j23373539.v5i2.16462

Hartanto, R., Kartikasari, A., 2016. Android based real-time static Indonesian sign language recognition system prototype, in: 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE). Presented at the 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE), pp. 1–6. https://doi.org/10.1109/ICITEED.2016.7863311

Hasan, M.A., Tolle, H., Brata, A.H., 2018. Implementasi Aplikasi Papan Komunikasi (Communication Board) Berbasis Tablet Android Bagi Tunarungu 7.

Khamid, Wibawa, A.D., Sumpeno, S., 2017. Gesture Recognition for Indonesian Sign Language Systems (ISLS) Using Multimodal Sensor Leap Motion and Myo Armband Controllers Based-on Naïve Bayes Classifier, in: 2017 International Conference on Soft Computing, Intelligent System and Information Technology (ICSIIT). Presented

at the 2017 International Conference on Soft Computing, Intelligent System and Information Technology (ICSIIT), pp. 1–6. https://doi.org/10.1109/ICSIIT.2017.42

Kuroki, K., Zhou, Y., Cheng, Z., Lu, Z., Zhou, Y., Jing, L., 2015. A remote conversation support system for deaf-mute persons based on bimanual gestures recognition using finger-worn devices, in: 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops). Presented at the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), pp. 574–578. https://doi.org/10.1109/PERCOMW.2015.7134101

Maulia, N., 2017. PENGARUH PENGGUNAAN SISTEM ISYARAT BAHASA INDONESIA (SIBI) TERHADAP PEMAHAMAN INFORMASI SISWA PENYANDANG TUNARUNGU DI SLB-PKK PROVINSI LAMPUNG 15.

Mohandes, M., Aliyu, S., Deriche, M., 2015. Prototype Arabic Sign language recognition using multi-sensor data fusion of two leap motion controllers, in: 2015 IEEE 12th International Multi-Conference on Systems, Signals Devices (SSD15). Presented at the 2015 IEEE 12th International Multi-Conference on Systems, Signals Devices (SSD15), pp. 1–6. https://doi.org/10.1109/SSD.2015.7348113

Nugroho, A.S., Witarto, A.B., Handoko, D., 2003. –Teori dan Aplikasinya dalam Bioinformatika1– 11.

Nuriyanti, Y., Tresnawati, D., 2015. PENGEMBANGAN APLIKASI PENGENALAN DASAR BAHASA ISYARAT SEBAGAI MEDIA PEMBELAJARAN BERBASIS ANDROID 12, 9.

Pramunanto, E., Sumpeno, S., Legowo, R.S., 2017. Classification of hand gesture in Indonesian sign language system using Naive Bayes, in: 2017 International Seminar on Sensors, Instrumentation, Measurement and Metrology (ISSIMM). Presented at the 2017 International Seminar on Sensors, Instrumentation, Measurement

and Metrology (ISSIMM), pp. 187–191. https://doi.org/10.1109/ISSIMM.2017.8124288

Saxena, A., Jain, D.K., Singhal, A., 2014. Hand Gesture Recognition Using an Android Device, in: 2014 Fourth International Conference on Communication Systems and Network Technologies. Presented at the 2014 Fourth International Conference on Communication Systems and Network Technologies, pp. 819–822. https://doi.org/10.1109/CSNT.2014.170

Sidabutar, R.P., Andhini, M., Daulika, C.A., Hidayati, H., Kom, S., 2015. Jurnal Aplikasi CallMe : Aplikasi Alat Bantu Komunikasi Jarak Jauh Untuk Penyandang Tuna Rungu Dan Penyandang Tuna netra 5.

Supria, Muhamad Nasir, R.F., 2017. AplikasiMarawis Digital Menggunakan Sensor Leap Motion.

Suyanto, 2017. DATA MINING, Mei 2017. ed, Untuk Klasifikasi dan Klasterisasi Data. Informatika Bandung.

Syafiq, M., Al Kadafi, A.J., Zakiyyah, R.H., Jauhari, D., Luqyana, W.A., Cholissodin, I., Muflikhah, L., 2016. Aplikasi Mobile (Lide) Untuk Diagnosis Tingkat Resiko Penyakit Stroke Menggunakan PTVPSO-SVM. Jurnal Teknologi Informasi dan Ilmu Komputer 3, 147. https://doi.org/10.25126/jtiik.201632190

Wibowo, M.D., Nurtanio, I., Ilham, A.A., 2017. Indonesian sign language recognition using leap motion controller, in: 2017 11th International Conference on Information Communication Technology and System (ICTS). Presented at the 2017 11th International Conference on Information Communication Technology and System (ICTS), pp. 67–72. https://doi.org/10.1109/ICTS.2017.8265648

Wibowo, M.D., Nurtanio, I., Ilham, A.A., n.d. Indonesian sign language recognition using leap motion controller - IEEE Conference Publication.

Wijayanto, C.P., 2009. MEMBANGUN APLIKASI PELATIHAN BAHASA ISYARAT BERBASIS KOMPUTER PADA ORANG TUNARUNGU.

LAMPIRAN 1

Biodata Guru Ahli Isyarat SIBI SLB Katolik Rajawali

Nama	: Dominika Roa,S.Pd
Tempat/Tgl Lahir	: Bokeka, 30 September 1994
No. Hp	: 0821919791446
NIP	: -
Alamat	: Jl. Lamadukelleng
Status Guru	: Guru Slb Kelas X
Lama Mengajar	: 1 Tahun 10 Bulan

LAMPIRAN 2

Dataset dari API *Leap Motion*

F51.1313374042511
aF15.881324768066406
aF-12.56293123960495
aF31.521901726722717
aF15.243303298950195
aF-86.32466745376587
aF11.343953967094421
aF16.304914474487305
aF-99.54169416427612
aF-6.881304860115051
aF18.875247955322266
aF-91.84874629974365
aF-28.42245662212372
aF13.700569152832031
aF-70.94187641143799
aF20.37058435165299
aF156.44460184473914
aa(lp22
F50.22144444783529
aF22.800189548068577
aF-10.649721569485134
aF30.59514586130778
aF22.468241373697918
aF-85.41178597344293
aF10.428424305386013

aF23.646447075737846
aF-98.40548027886285
aF-9.055412451426188
aF28.051076253255207
aF-88.82417085435655
aF-29.445795483059353
aF22.36202833387587
aF-68.45867029825847
aF17.967585751187087
aF153.2526798963134
aa(lp23
F50.776134729385376
aF18.297447204589844
aF-12.370995223522186
aF31.58326506614685
aF18.664703369140625
aF-86.30322408676147
aF11.083371877670288
aF19.816783905029297
aF-99.37541055679321
aF-6.3777996301651
aF18.398956298828125
aF-92.52093887329102
aF-29.20080268383026

aa(lp23
F50.776134729385376
aF18.297447204589844
aF-12.370995223522186
aF31.58326506614685
aF18.664703369140625
aF-86.30322408676147
aF11.083371877670288
aF19.816783905029297
aF-99.37541055679321
aF-6.3777996301651
aF18.398956298828125
aF-92.52093887329102
aF-29.20080268383026
aF15.143062591552734
aF-71.31683111190796
aF17.774632005928712
aF152.72340720943677
etc

LAMPIRAN 3

Source Code pada proses penerjemah

```
import math
class SVM:
	def __init__(self):
		self.classes = {}
	def train(self, kernel):
		"""Given the training features X with labels y, returns a SVM
    predictor representing the trained SVM.
    """
		# Sort kernel by class
		classesData = {}
		for labels in kernel:
			# If class is new
			if labels[0] not in classesData:
				classesData[labels[0]] = [labels[1]]
				self.classes[labels[0]] = []

			# Append to existing class
			else:
				classesData[labels[0]].append(labels[1])

		# Compute formula components per class
		for _class, linear in classesData.iteritems():
			prior = math.log(len(linear)/float(len(kernel)))
			mu = []
			variance = []

			# Fill lists of mu and variance for each feature
			for feature in zip(*linear): # * selects columns instead of rows
				avg = sum(feature)/float(len(feature))
				mu.append(avg)
				variance.append(sum(map(lambda x : (x - avg) ** 2, feature))
/ len(feature)) #

			# Precompute first part of formula
			base = sum(map(lambda x : math.log(2 * math.pi * x), variance)) * (-
0.5)

			maxDistance = 0
			totalDistance = 0
			for sample in linear:
				euclid = self.euclidian(sample, mu)
				totalDistance += euclid
				if euclid > maxDistance:
					maxDistance = euclid
```

```
# Store results
                        self.classes[_class] = (mu, variance, prior + base, maxDistance)
        def euclidian(self, pVector, qVector):
                '''
                Get euclidian distance between vector p and q
                '''
                return math.sqrt(sum([(p - q) ** 2 for p, q in zip(pVector, qVector)]))
        def support_vectors(self, vector):
                '''
                Get the support vectors for all classes
                along with euclidian distances
                '''
                support_vectors = []
                distances = [] #euclidian distances for novelty detection

                for _class in self.classes:
                        _vars = zip(vector, self.classes[_class][0], self.classes[_class][1])

                        # Calculate rest of formula; the exponent
                        exp = (-0.5) * sum(map(lambda x : (x[0] - x[1])**2 / x[2], _vars))
                        support_vectors.append(self.classes[_class][2]+exp)

                        # Calculate how many max distances vector is from average in n-
dimensions
                        distances.append(self.euclidian(vector, self.classes[_class][0]) /
self.classes[_class][3])
                # Normalize; convert log between 0 and 1
                _sum = sum(support_vectors)
                output = []
                for _class, support_vector_labels, dist in zip(self.classes, support_vectors,
distances):
                        output.append((_class, (1 - (support_vector_labels / _sum)) / 2.0,
dist))
                return sorted(output, key=lambda x: x[1], reverse=True)

        def classify(self, vector):
                '''
                Get name only of the best candidate of class match
                '''
                return self.support_vectors(vector)[0][0]
        def accuracy(self, kernel):
                '''
                Test accuracy of trained classifier
                '''
                return sum([d[0] == self.classify(d[1]) for d in kernel]) / float(len(kernel))
```