

University at Buffalo
CSE 574 Introduction to Machine Learning

Programming Assignment 2 Report

Submitted by: **Group 95**
Vishal Chaurasia – 50353581
Pushkar K Pandey – 50412529
Varun Sudarshan – 50360324

November 14, 2021

Introduction

In this assignment, we had to implement a Multilayer Perceptron Neural Network & evaluate its performance in classifying handwritten digits. Additionally, we used the same network to analyze a more challenging face dataset and compare the neural network's performance against a deep neural network and a convolutional neural network using the TensorFlow library.

Feature Selection

While pre-processing the dataset, we noticed many redundant features, i.e., their values were the same for all points in the training set. Since no additional information could be derived from the redundant features, we choose to ignore them.

Total number of features selected: **717**.

The following figure depicts the indices of the features selected.

```
12 13 14 15 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 86 87 88 89 90 91 92 93 94 95 96
97 98 99 100 101 102 103 104 105 106 107 108 109 110 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 142 143 144 145
146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190
191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278
279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322
323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366
367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410
411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454
455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499
500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543
544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588
589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632
633 634 635 636 637 638 639 640 641 642 643 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 674 675 676 677 678 679 680 681
682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 731 732
733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779
Total Selected Features-->717
preprocess done
```

How to decide hyper-parameter for Neural Network:

As explained below, we achieved the best combination by using different values of λ and no of hidden units (m) and comparing the test accuracy. We found the optimal hyperparameters (regularization coefficient, λ and no of hidden units m) combination with supporting data.

Choosing the best combination:

Ranging λ from 0-60 with an increment value of 10 & m from 4-20 with an increment value of 14:

λ	m	Train Accuracy	Validation accuracy	Test accuracy	Training Time
30	20	93.53	92.820	93.300	34.67
00	20	93.528	92.39	93.28	34.73
00	16	93.66	92.61	93.28	30.93
10	20	93.62	93.02	92.91	32.23
10	12	91.67	90.77	91.5	30.12
20	20	93.226	92.528	92.77	34.28
30	60	92.476	91.73	92.14	30.65

Above are the top hyperparameter combination that resulted in the best testing accuracy result

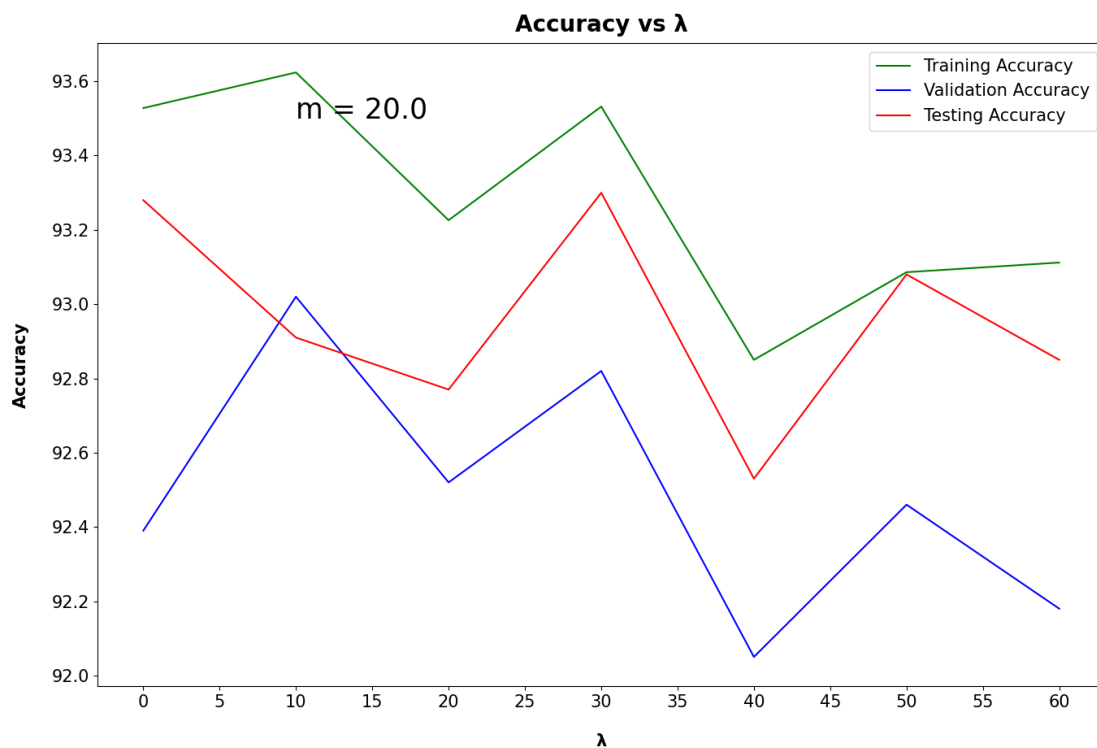
We can notice from the above table that our optimum λ and m combination is 30 and 20; this gives us the best test accuracy, 93.30 %.

We try changing λ by fixing the value of m to 20.

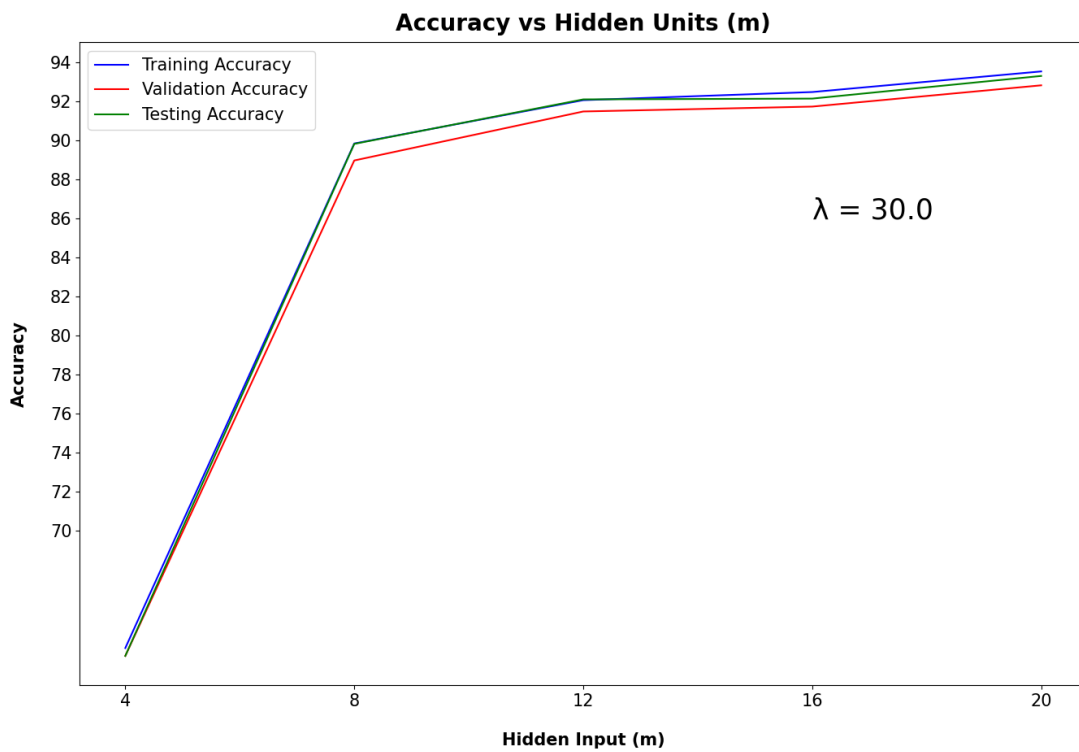
We can find from the resulting table that the value of $\lambda = 30$ gives us the optimum value.

λ	m	Train Accuracy	Validation accuracy	Test accuracy	Training Time
00	20	93.528	92.39	93.28	34.73
10	20	93.624	93.02	92.91	32.23
20	20	93.226	92.52	92.77	34.28
30	20	93.53	92.82	93.30	34.67
40	20	92.85	92.05	92.53	33.11
50	20	93.08	92.46	93.08	32.51
60	20	93.11	92.17	92.85	36.51

From the below graph, we can say that we get the best testing accuracy at $\lambda = 30$.



We try changing m by fixing the value of λ to 30.



We can find from the resulting table that the value of $m = 20$ gives us the optimum value.

λ	m	Train Accuracy	Validation accuracy	Test accuracy	Training Time
30	04	63.99	63.59	63.59	26.15
30	08	89.84	88.97	89.82	27.11
30	12	92.06	91.47	92.10	28.15
30	16	92.47	91.73	92.14	30.65
30	20	93.53	92.82	93.30	34.67

By taking Hyper Parameter combination: $\lambda = 30$, $m = 20$ we get the best test accuracy as 93.30%

Accuracy on the celebA dataset:

We reused our implementations of the functions - sigmoid(), nnObjFunc() and nnPredict() from the nnScript.py in the facennScript.py. Below are the accuracy obtained by running the facennScript.py for $\lambda = 10$:

<u>Training set Accuracy:</u>	84.22%
<u>Validation set Accuracy:</u>	83.23%
<u>Test set Accuracy:</u>	84.44%
<u>Training Time:</u>	158.43 s

Deep NN Observations:

Hidden Layers	Accuracy (%)	Training time (s)
3	81.33	465.05
5	81.86	438.45
7	82.70	462.07
9	83.34	543.26
11	80.58	399.01

- The accuracy increases with an increase in the number of hidden layers up to a certain point and then decreases due to overfitting
- The Peak Accuracy is 83.34%
- The peak accuracy is less than the accuracy we observed for a NN with one hidden layer (84.44%).
- The training time for the NN (158.43 s) was significantly lesser than the deep NN's training time as there is only 1 hidden layer.

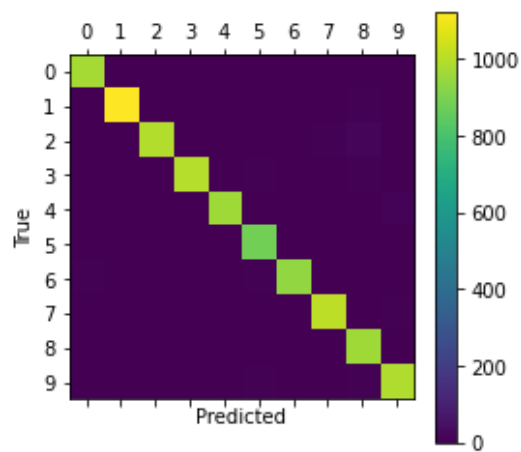
CNN Observations:

Time usage: 0:11:04

Accuracy on Test-Set: 98.5% (9847 / 10000)

Confusion Matrix:

```
[[ 973    0    0    0    0    1    2    1    3    0]
 [    0 1124    1    0    0    0    1    1    8    0]
 [    4    4  995    4    1    0    0    5   19    0]
 [    1    0    0  997    0    5    0    2    5    0]
 [    1    0    1    0  965    0    1    2    1   11]
 [    2    0    0    2    0  885    1    0    2    0]
 [    5    1    0    1    1    7  942    0    1    0]
 [    0    1    3    1    0    0    0 1014    2    7]
 [    3    0    1    2    0    3    0    0  963    2]
 [    1    3    0    1    2    5    1    2    5  989]]
```



- Accuracy of CNN is 98.5 percent
- Training Time: 11 Minutes 04 Seconds