文章编号：2095-2163(2023)07-0058-06 中图分类号：TP391.41 文献标志码：A

基于遗传算法求解 TSP 问题的研究及 Matlab 实现

杨锦涛，赵春香，杨成福
（云南师范大学 信息学院，昆明 650500）

摘 要：TSP 问题属于组合优化问题，同时也是一个 NPC 问题，因此人们一直致力于为其寻找有效的近似求解算法。遗传算法是模仿生物进化而构建的一种随机搜索方法，具有较强的全局搜索能力、潜在的并行性以及良好的可扩展性，能有效求解 TSP 问题。然而，如何确定遗传参数和选择遗传操作一直是一个难题，本文针对 TSP 问题的求解构建完整的遗传算法体系，选择合适的参数，设计多组交叉算子和变异算子，分别对 TSP 问题进行求解。通过多次实验以及对实验结果的分析比较，探究不同的交叉算子和变异算子求解 TSP 问题的效果，为遗传操作中交叉算子和变异算子的选择提供一定的参考。

关键词：TSP 问题；组合优化；遗传算法

Research on solving TSP problem based on genetic algorithm and Matlab implementation

YANG Jintao, ZHAO Chunxiang, YANG Chengfu
(School of Information, Yunnan Normal University, Kunming 650500, China)

【Abstract】 The TSP problem belongs to combinatorial optimization problems and is also an NPC problem, so people have been trying to find the corresponding effective approximation solving algorithms. Genetic algorithm is a random search method built to imitate biological evolution, with strong global search ability, potential parallelism and good scalability, which can effectively solve TSP problems. However, how to determine genetic parameters and select genetic manipulation has always been a difficult problem. In this paper, a complete genetic algorithm system is constructed for the solution of TSP problems, appropriate parameters are selected, and multiple sets of cross operators and mutation operators are designed to solve TSP problems separately. Through multiple experiments and the analysis and comparison of experimental results, the effect of different crossover operators and mutation operators in solving TSP problems is explored, which provides a certain reference for the selection of crossover operators and mutation operators in genetic operations.

【Key words】 TSP problem; combinatorial optimization; genetic algorithm

0 引 言

TSP 问题、即巡回旅行商问题，是组合优化领域中的一个典型问题。现实生活中的很多实际应用问题都可以简化为 TSP 问题。TSP 问题可以用图论描述为：已知带权完全图 G，求一条使得路径总和最小、且经过所有顶点的回路。TSP 问题虽然描述简单、容易理解，但是求解是很困难的。当问题的规模较小时，仅使用枚举法就能找到一条最优路径，但当城市数量较多时，即使用计算机也无法将解全部列举，要求出 TSP 问题的最优解是不可能的。

遗传算法是一种自组织、自适应的全局寻优算法，因其潜在的并行性、较高的鲁棒性，在应用研究方面取得了很多可观的成果，被广泛应用于函数优化、组合优化、生产调度、自适应控制、图像处理、机器学习、数据挖掘、人工生命、遗传编程等领域。

1975 年，Holland[1] 受生物学中生物进化和自然选择学说的启发，提出了著名的遗传算法。2006 年，何燕[2] 对遗传算法进行改进，将其应用到车间调度领域。2010 年，蒋波[3] 将遗传算法应用于车辆路径优化问题，指出遗传算法求解该问题的优越性，并对其做出了改进，实验证明改进后的遗传算法能

基金项目：云南师范大学博士科研启动基金(2021ZB019)。

作者简介：杨锦涛(2002-)，女，本科生，主要研究方向：深度学习；赵春香(2000-)，女，本科生，主要研究方向：深度学习；杨成福(1986-)，男，博士，讲师，硕士生导师，主要研究方向：信息超材料、深度学习、人工智能。

通讯作者：杨成福 Email：yangchengfu@ynnu.edu.cn

收稿日期：2022-12-18

够有效解决此类问题。2013 年，乔阳[4]将遗传算法和 Ostu 图像分割法进行改进后结合在一起进行图像分割实验，得到了满意的结果。

遗传算法是模拟生物进化的过程发展而来的一种算法，从一定规模的解集（初始种群）开始，通过选择、交叉和变异，将适应度低的解（个体）淘汰掉，将适应度高的解（个体）保留下来，并产生新的解（个体），生成新的解集（种群），通过不断地迭代，使解集（种群）中的解（个体）越来越接近问题的最优解。生物学和遗传算法概念之间的对应关系见表 1。

表 1　生物学和遗传算法概念对照

Tab. 1　Comparison of biological and genetic algorithm concepts

生物学	遗传算法
外界环境	约束条件
个体	问题的一个可行解
个体对环境的适应度	可行解的质量
种群	一定数量可行解的集合
生物的繁衍	算法的迭代
种群的进化过程	可行解的优化过程

本文针对 TSP 问题构建完整的遗传算法体系，将求解 TSP 问题几种常用的交叉算子和变异算子两两组合在一起，分别对具体的 TSP 问题实例进行求解，从所得的最优解和求解的时间两方面对实验结果进行分析和总结，探究使用不同的交叉算子和变异算子时遗传算法求解对称式 TSP 问题的效果[5]。

1　遗传算法求解 TSP 问题

1.1　编码

编码是指按照一定的构造方法，将问题的可行解转变为遗传算法能直接处理的个体。常用的编码方式有二进制编码、近邻编码、次序编码、路径编码[6]。

使用路径编码求解 TSP 问题，不仅编码过程简单易操作，而且编码结果非常直观，即首先对城市进行编号，然后以城市编号作为城市的编码，因此本文选择使用路径编码方式对城市进行编码。

1.2　初始种群

一般地，初始种群采用随机方法生成，如果种群规模为 M，则随机生成 M 个个体。

1.3　适应度函数

适应度函数用于对种群中的个体进行优劣程度的评价，由于算法在搜索最优解的过程中主要以个体的适应度作为依据，所以如果适应度函数构建不当，很可能导致算法的收敛速度缓慢、甚至无法收敛，即适应度函数直接决定着算法的收敛能力和寻优能力。

对于 TSP 问题，适应度函数一般取路径总和的倒数，具体定义公式见如下：

$$f(x)=\frac{1}{\sum_{i=1}^{n-1} d(t_i,t_{i+1})+d(t_n,t_1)} \tag{1}$$

其中，n 表示城市的数量；$T=(t_1,t_2,\cdots,t_n)$ 为种群中的一个个体；$d(t_i,t_j)$ 表示城市 i 到城市 j 的距离。TSP 问题为最小值问题，由适应度函数可知，路径总和与个体适应度呈倒数关系。

1.4　遗传操作

1.4.1　选择算子

选择是用选择算子对个体进行筛选的过程，这一过程中，差的个体被保留下来的概率小，好的个体被保留下来的概率大，会使种群中的个体向最优解进化。常用的选择算子有轮盘赌选择、最佳个体保存选择、锦标赛选择和排序选择[7]。

轮盘赌选择是 TSP 问题求解最常用的选择算子，即使用适应度值计算出每个个体被选择的概率，并根据该概率值对种群中的个体进行选择。本文也使用轮盘赌选择作为选择算子，并在轮盘赌的基础上添加最佳个体保存选择，即把种群中出现过的适应度值最高的个体保留下来，避免种群中优秀的个体在遗传操作中被淘汰或破坏。

1.4.2　交叉算子

个体交叉是为了实现种群的更新，而交叉算子是进行交叉的手段，定义了个体之间以怎样的方式交叉。对于不同问题，由于编码方式的不同，交叉算子也有所不同。对此拟做研究分述如下。

（1）部分匹配交叉（PMX）：首先采用随机方式在父体中确定 2 个位置，由 2 个位置确定一个交叉段，然后将 2 个父体的交叉段进行交换，最后根据交叉段之间的映射关系消除子代中的重复基因。

（2）顺序交叉（OX）：首先从父体中随机选择 2 个位置，由 2 个位置确定一个基因段，然后将父体 A 的该基因段复制到子代 A' 的对应位置，最后将父体 B 除父体 A 被选择的基因段之外的基因依次复制到子代 A' 的其余位置，同理可得到子代 B'。

（3）循环交叉（CX）：首先将父体 A 的第一个基因复制到子代，然后在父体 B 中的相同位置查看基因，随后在父体 A 中找到该基因复制到子代的相同位置，并在父体 B 中查看相同位置的基因，重复此步骤，直到在父体 B 中找到的基因已经在子代中，停止循环，在父体 B 中找到剩余的基因，并按照顺序复制到子代中的剩余位置。

1.4.3 变异算子

变异操作的主要目的是维持种群多样性,在遗传算法后期,个体交叉产生新个体的能力弱,通过个体变异可以进一步产生新个体,扩大搜索空间。接下来给出剖析论述如下。

(1)对换变异。首先用随机方式在父体中确定 2 个位置,然后交换这 2 个位置上的基因。

(2)倒位变异。用随机方式在父体中确定 2 个位置,以确定一个基因段,然后将其进行逆序排列。

(3)插入变异。用随机方式在父体中确定一个位置,以确定一个待插入的基因,再用随机方式确定 2 个位置,以确定插入点,最后将待插入的基因放入插入点。

1.5 遗传算法求解 TSP 问题具体步骤

根据上文选择的实现技术构建完整的遗传算法体系后,对 TSP 问题实例进行求解的具体步骤如下:

Step 1 获取城市数据,对城市进行编号。

Step 2 初始化种群。

Step 3 适应度评价。

Step 4 执行选择操作,采用轮盘赌选择对个体进行筛选,选出足够数量的个体。

Step 5 执行交叉操作,将选择操作中选出的个体两两组合作为父染色体,判断是否进行交叉,如果进行交叉,则按照选定的交叉算子进行交叉。

Step 6 执行变异操作,将执行交叉操作后的每个个体作为父染色体,判断是否进行变异,如果进行变异,则按照选定的变异算子进行变异。

Step 7 完成变异后,执行最佳个体保存策略,判断当前种群中的最优解是否优于历史最优解。如果是,更新历史最优解,否则找出种群中最差的解,用最优解将其替换掉。

Step 8 判断是否继续进行迭代,若是,回到 Step3;否则,结束迭代,输出最优解。

将前述的 3 种交叉算子和 3 种变异算子两两组合在一起,共有 9 种组合方式,见表 2。基于表 2 中列出的 9 种组合,重复对 TSP 问题进行求解。

表 2 9 组交叉算子和变异算子

Tab. 2 9 sets of crossover and mutation operators

组合编号	交叉算子	变异算子
第一组	部分匹配交叉	对换变异
第二组	部分匹配交叉	倒位变异
第三组	部分匹配交叉	插入变异
第四组	循序交叉	对换变异
第五组	循环交叉	倒位变异
第六组	循环交叉	插入变异
第七组	顺序交叉	对换变异
第八组	顺序交叉	倒位变异
第九组	顺序交叉	插入变异

2 实验及结果分析

2.1 实验仿真

本文使用 Matlab 实现上文构建的遗传算法,从 TSPLIB 中选择测试样例进行具体分析。TSPLIB 是包含对称旅行商问题、哈密顿回路问题、以及非对称旅行商问题的多种实例数据的文件库,数据规模多样。本文在对每个测试样例进行求解时,改变算法的交叉算子和变异算子,进行多次重复的实验,从问题的最优解和求解时间两方面对几组交叉算子和变异算子求解 TSP 问题的效果进行分析比较。

在完成算法的编程后,初步设置参数,对实例 Oliver30 进行求解,根据实验结果对算法的参数进行调整,最终选定迭代次数 G 为 500,交叉概率 Pc 为 0.9,变异概率 Pm 为 0.2,种群规模 M 根据待求解问题的规模来确定。一般地,城市个数越多,种群规模越大,对 30 个城市的实例 Oliver30,种群规模取 100。用上述 9 组交叉算子和变异算子分别对 Oliver30 求解 10 次的结果见表 3。

表 3 遗传算法求解 Oliver30

Tab. 3 Genetic algorithm for Oliver30

序号	第一组	第二组	第三组	第四组	第五组	第六组	第七组	第八组	第九组
1	511.98	425.10	526.40	510.03	425.48	496.34	532.75	425.72	472.36
2	565.56	429.62	451.69	538.10	423.74	479.01	501.09	428.04	495.35
3	495.80	432.66	468.32	496.65	476.12	495.19	514.30	428.84	476.09
4	496.84	425.10	433.09	534.14	424.12	457.73	501.54	431.71	457.74
5	541.80	460.17	464.91	536.89	451.72	445.97	496.24	435.31	474.21
6	500.88	429.83	494.45	525.44	438.38	483.92	507.57	450.41	478.80
7	501.92	432.66	521.33	514.30	425.27	466.65	561.85	428.46	460.74
8	514.90	450.69	497.46	488.21	425.31	484.18	515.13	423.74	474.73
9	529.06	447.21	487.60	482.61	434.61	500.18	571.79	431.76	466.69
10	556.31	425.73	507.64	510.61	433.54	471.18	456.79	438.38	483.27

以文献[8]中求得的最优解 423.74 作为参考值，从表 3 可以看出第 2 组、第 5 组、第 8 组交叉算子和变异算子的求解结果都比较接近参考最优解，且较稳定，说明参数设置较合理。其中一次求解的收敛曲线如图 1 所示，最优路线如图 2 所示。其中，以圆圈标记的点为路线起点，其与路线终点用虚线相连，其余路线用实线连接。

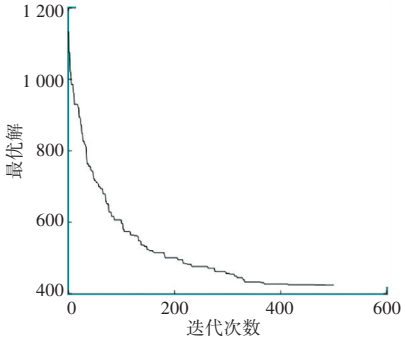

图 1　Oliver30 收敛曲线

Fig. 1　Convergence curve of Oliver30

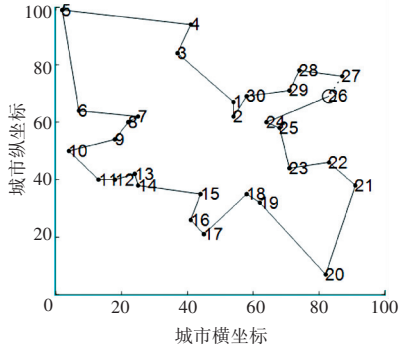

图 2　Oliver30 最优路线图

Fig. 2　Optimal roadmap of Oliver30

2.2　实验结果

从 TSPLIB 中选择测试样例进行求解，本文总共选择了 5 个实例，分别是 ulysses16、dantzig42、eil51、eil76、eil101，根据问题的规模为每个实例设置合适的种群规模（*M*）。其中，ulysses16、dantzig42、eil51 实例的种群规模取 100，eil76 实例的种群规模取 150，eil101 实例的种群规模取 200，分别用上述 9 组交叉算子和变异算子求解 10 次，记录 10 次求解结果的最好值（*Best*）、平均值（*AVR*）和偏差率（*Dr*），以及求解的平均时间（*Time*），这里的偏差率可由式(2)来计算：

$$Dr = \frac{Best - Opt}{Opt} \tag{2}$$

其中，*Opt* 是 TSPLIB 数据集提供的最优解。实验结果见表 4～表 8。

表 4　遗传算法求解 ulysses16

Tab. 4　Genetic algorithm for ulysses16

序号	*AVG*	*Best*	*Opt*	*Dr*	*Time*/ s
第 1 组	75.01	74.00		0.00	1.2
第 2 组	74.01	73.99		0.00	1.1
第 3 组	74.24	73.99		0.00	1.4
第 4 组	74.36	73.99		0.00	0.7
第 5 组	74.05	73.99	74	0.00	0.7
第 6 组	74.20	73.99		0.00	0.7
第 7 组	74.60	73.99		0.00	0.5
第 8 组	74.09	73.99		0.00	0.5
第 9 组	74.28	73.99		0.00	0.5

表 5　遗传算法求解 dantzig42

Tab. 5　Genetic algorithm for dantzig42

序号	*AVG*	*Best*	*Opt*	*Dr*	*Time*/ s
第 1 组	1 022.34	864.89		0.24	1.4
第 2 组	756.49	713.99		0.02	1.6
第 3 组	915.69	838.38		0.20	2.8
第 4 组	944.02	858.26		0.23	1.0
第 5 组	733.00	698.98	699	0.00	1.0
第 6 组	894.32	794.02		0.14	1.0
第 7 组	1 061.21	893.35		0.28	0.7
第 8 组	760.73	725.37		0.04	0.6
第 9 组	937.22	846.41		0.21	0.7

表 6　遗传算法求解 eil51

Tab. 6　Genetic algorithm for eil51

序号	*AVG*	*Best*	*Opt*	*Dr*	*Time*/ s
第 1 组	642.66	613.87		0.44	1.4
第 2 组	516.35	487.95		0.15	1.8
第 3 组	592.00	532.90		0.25	3.6
第 4 组	646.09	559.68		0.31	1.1
第 5 组	503.97	481.67	426	0.13	1.1
第 6 组	580.39	534.41		0.25	1.1
第 7 组	657.48	604.81		0.42	0.7
第 8 组	511.91	465.28		0.09	0.7
第 9 组	607.42	551.70		0.30	0.7

表 7 遗传算法求解 eil76

Tab. 7 Genetic algorithm for eil76

序号	*AVG*	*Best*	*Opt*	*Dr*	*Time*/ s
第 1 组	1 037.94	971.96		0.81	2.2
第 2 组	880.64	806.53		0.50	3.2
第 3 组	1 002.01	916.52		0.70	9.5
第 4 组	1 012.32	949.17		0.76	1.8
第 5 组	835.37	792.39	538	0.47	1.8
第 6 组	993.69	925.74		0.72	1.8
第 7 组	1 073.78	1 014.86		0.89	0.9
第 8 组	833.77	781.88		0.45	1.0
第 9 组	992.89	894.47		0.66	1.0

表 8 遗传算法求解 eil101

Tab. 8 Genetic algorithm for eil101

序号	*AVG*	*Best*	*Opt*	*Dr*	*Time*/ s
第 1 组	1 470.94	1 361.78		1.16	3.2
第 2 组	1 340.33	1 286.64		1.05	5.3
第 3 组	1 452.02	1 363.67		1.17	20.4
第 4 组	1 468.17	1 352.24		1.15	2.7
第 5 组	1 194.49	1 142.19	629	0.82	2.7
第 6 组	1 408.33	1 297.43		1.06	2.7
第 7 组	1 479.12	1 397.69		1.22	1.2
第 8 组	1 238.68	1 168.25		0.86	1.2
第 9 组	1438.39	1 340.93		1.13	1.2

为了方便对比,将每个实例求解结果中的偏差率(*Dr*)和求解的平均时间(*Time*)分别统计在一起,由于实例 ulysses16 问题规模小,坐标数据也容易处理,不管选择哪种交叉算子和变异算子,求解结果都很接近最优解,因此在进行偏差率的比较时不将其考虑在内,具体见表 9、表 10。

表 9 偏差率对比

Tab. 9 Deviation rate comparison

序号	dantzig42	eil51	eil76	eil101
第 1 组	0.24	0.44	0.81	1.16
第 2 组	0.02	0.15	0.50	1.05
第 3 组	0.20	0.25	0.70	1.17
第 4 组	0.23	0.31	0.76	1.15
第 5 组	0.00	0.13	0.47	0.82
第 6 组	0.14	0.25	0.72	1.06
第 7 组	0.28	0.42	0.89	1.22
第 8 组	0.04	0.09	0.45	0.86
第 9 组	0.21	0.30	0.66	1.13

表 10 求解平均时间对比

Tab. 10 Comparison of average solving time

序号	ulysses16	dantzig42	eil51	eil76	eil101
第 1 组	1.2	1.4	1.4	2.2	3.2
第 2 组	1.1	1.6	1.8	3.2	5.3
第 3 组	1.4	2.8	3.6	9.5	20.4
第 4 组	0.7	1.0	1.1	1.8	2.7
第 5 组	0.7	1.0	1.1	1.8	2.7
第 6 组	0.7	1.0	1.1	1.8	2.7
第 7 组	0.5	0.7	0.7	0.9	1.2
第 8 组	0.5	0.6	0.7	1.0	1.2
第 9 组	0.5	0.7	0.7	1.0	1.2

3 实验结论

根据上述实验结果,可以得出如下结论:

(1)表 9 中的偏差率描述了采用不同的交叉算子和变异算子时,所求得的最优解与 TSPLIB 中给出的最优解的差距,偏差率越小,说明算法求得的结果越接近最优解,算法的寻优能力越好。从表 9 中可以看出,第 2 组数据总是小于第 1 组和第 3 组、第 5 组数据总是小于第 4 组和第 6 组、第 8 组数据总是小于第 7 组和第 9 组,这说明每种交叉算子和逆转变异组合在一起时,问题的求解结果总是比与对换变异和插入变异组合在一起时更接近最优解。由此可知,遗传算法使用逆转变异作为变异算子时比选择对换变异和插入变异作为变异算子的寻优能力更强。

(2)表 10 是采用每组交叉算子和变异算子求解每个实例 10 次所花时间的平均值,所花的时间越少,说明算法的搜索速度越快,执行效率越高,由于变异操作比较简单,所以遗传算法的执行效率主要由交叉操作决定。从表 10 中可以看出,遗传算法采用顺序交叉和循环交叉时,即使采用不同的变异算子,所花的时间也基本相同,但是采用部分匹配交叉所花的时间会因为变异算子的不同而有所不同。对于每一个实例,在得到的解的质量差别不大的情况下,遗传算法使用部分匹配交叉所花的时间最多,使用循环交叉所花的时间最少。

综上所述,对于比较简单的 TSP 问题,由于使用遗传算法总能求得与最优解很接近的解,所以选择何种交叉算子和变异算子对算法的寻优能力影响不大,但是使用部分匹配交叉会花费比较多的时间,会导致算法的执行效率低,因此交叉算子选择循环

交叉比较合适。对于不是总能求得最优解的 TSP 问题，与对换变异和插入变异相比，使用逆转变异会使算法具有更强的寻优能力，找到的最优解更接近最优解，使用部分匹配交叉和顺序交叉会花费比循环交叉更多的时间，使算法的执行效率变低，而且找到的最优解也不会更优。

4　结束语

本文对几种常用的交叉算子和变异算子求解 TSP 问题的效果进行了研究。实验结果表明，在几种常用的交叉算子和变异算子中，选择循环交叉和逆转变异算法的执行效率最高，寻优能力最好，这能为遗传算法中交叉算子和变异算子的选择提供一定的参考，同时有利于设计出更好的交叉算子和变异算子，提高算法的性能。

参考文献

[1] HOLLAND J. Adaptation in natural and artificial systems: an introductory analysis with application to biology[J]. Control Artificial Intelligence, 1975.

[2] 何燕. 基于遗传算法的车间调度优化及其仿真[D]. 武汉:武汉理工大学, 2006.

[3] 蒋波. 基于遗传算法的带时间窗车辆路径优化问题研究[D]. 北京:北京交通大学, 2010.

[4] 乔阳. 基于改进遗传算法的图像分割方法[D]. 成都:电子科技大学, 2013.

[5] 张家善, 王志宏, 陈应显, 等. 一种求解旅行商问题的改进遗传算法[J]. 计算机系统应用, 2012, 21(09): 192-194,191.

[6] 王娜. 求解 TSP 的改进遗传算法[D]. 西安:西安电子科技大学, 2010.

[7] 于丰瑞. 基于改进的遗传算法求解 TSP 问题[D]. 呼和浩特:内蒙古农业大学, 2016.

[8] 闫茹. 基于改进遗传算法的旅游路线优化研究与应用[D]. 银川:北方民族大学, 2021.

(上接第 57 页)

[3] 田浩杰,杨晓庆,翟晓雨. 基于深度学习的线圈炮缺陷自动检测与分类[J]. 现代计算机,2022,28(10):86-91.

[4] 张浩,吴陈,徐影. 基于深度学习在海缆表面缺陷检测中的应用[J]. 电脑知识与技术,2022,18(15):88-91.

[5] 陈宗仁,谢文达,余君,等. 基于深度学习的金属机械零件表面缺陷检测方法[J]. 制造业自动化,2021,43(12):170-173.

[6] 王昊,李俊峰. 基于深度学习的车载导航导光板表面缺陷检测研究[J]. 软件工程,2022,25(03):34-38,16.

[7] 刘瑞珍,孙志毅,王安红,等. 基于深度学习的偏光片缺陷实时检测算法[J]. 太原理工大学学报,2020,51(01):125-130.

[8] 王鸣霄,范娟娟,周磊,等. 基于深度学习的排水管道缺陷自动检测与分类[J]. 给水排水,2020,46(12):106-111.

[9] 施恺杰,王颖,王嘉璐,等. 基于深度学习的电子换向器表面缺陷检测[J]. 网络安全技术与应用,2021(06):113-115.

[10] 于宏全,袁明坤,常建涛,等. 基于深度学习的铸件缺陷检测方法[J]. 电子机械工程,2021,37(06):59-64.

[11] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[12] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84-90.

[13] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[14] SZEGEDY C, LIU Wei, JIA Yanqing, et al. Going deeper with convolutions[C]// IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, MA, USA: IEEE, 2015: 1-9.

[15] HE Kaiming, ZHANG Xiangyu, REN Shaoqing, et al. Deep residual learning for image recognition[C]// IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE, 2016: 770-778.